موسسه انتشارات بعثت

زبان برنامه نویسی
BASIC
برای کامپیوترهای کوچک

نویسنده :
جف می نارد

مترجم :
غلامرضا آذری



مؤسسه انتشارات بعثت

تلفن‌های فروشگاه و نمایشگاه مرکزی انتشارات بعثت و شعبات تابعه

– فروشگاه مرکزی و شماره ۱: خیابان انقلاب خیابان ۱۶ آذر پلاک ۶۸ تلفن: ۹۳۹۹۹۹

– فروشگاه شماره ۲: خیابان انقلاب نرسیده به چهارراه فلسطین تلفن: ۶۴۴۵۸۱–۶۴۶۶۶۱

– انبار: خیابان سعدی شمالی– خیابان منوچهری– نرسیده به چهارراه کنت پلاک ۵۸ تلفن: ۳۰۳۲۰۳

بسم الله الرحمن الرحیم

زبان برنامه نویسی
بیسیک
برای کامپیوترهای کوچک
BASIC

نوشته: جف می‌نارد
ترجمه: غلامرضا آذری

مؤسسه انتشارات بعثت

• نام کتاب: زبان برنامه‌نویسی
«بیسیک»
برای کامپیوترهای کوچک

• مترجم: غلامرضا آذری

• ناشر: مؤسسه انتشارات بعثت

• نوبت چاپ: **چاپ دوم**

• تعداد: **۳۰۰۰ جلد**

• تاریخ انتشار: **آذرماه ۱۳۶۶**

• **چاپ وصحافی سعیدنو**

• **فیلم‌وزینگ ــ لیتوگرافی اشکان**

سخن ناشر:

مؤسسه غیر انتفاعی انتشارات بعثت با همه محدودیتهائی که در رژیم منفور گذشته در نشر فرهنگ و معارف اسلامی داشته با تلاش پی‌گیر بیش از ۲۰ ساله خود بحمدالله توانسته کتب ارزنده دینی و علمی و اخلاقی به‌جامعه اسلامی تقدیم کند که پاره‌ای از آنها امروز نایاب و تعدادی کمیاب است که انشاءالله با تجدید چاپ آنها نیاز علاقمندان بزودی مرتفع خواهد شد.

امید است با شرایط سهل و آسانی که انقلاب شکوهمند اسلامی در نشر آثار علمی و دینی فراهم آورده. همچنان با استعانت از خداوند متعال و ائمه طاهرین(ع) راهی را که سالها پیش در آن گام نهاده‌ایم ادامه دهیم و به‌فراخور خویش دین خود را به دینی که برترین آئین است ادا نمائیم.

اینک فهرستی از آخرین آثار منتشره از سال ۱۳٦٤ تا ۱۳٦٦ را به‌علاقمندان تقدیم میداریم.

جلدهای ۱ تا ۵ دائرة الفرائد: تألیف دکتر محمدباقر محقق

قرآن و امام حسین(ع): نوشته دکتر محمدباقر محقق

دو زن و یکمرد: نوشته شهید بنت‌الهدا صدر
ترجمه سید جمال‌الدین حکیمی زاده

بیگانگی در باختر زمین: نوشته شهاب‌الدین سهروردی
بازگردان به‌خامهٔ دکتر حسین خراسانی

پنج مقاله: نوشته فخرالدین حجازی

بحثی درباره صلح امام حسن(ع): نوشته غلامرضا اشرف سمنانی

روش شوهرداری: نوشته دکتر شهید پاکنژاد

خواب و بیداری: نوشته دکتر محمدباقر محقق

تحلیلی از مدیریت اسلامی: نوشته غلامرضا اشرف سمنانی

خدا و علم: ترجمه علی‌اکبر کسمایی

داستانهایی از لابلای تاریخ: ترجمه و تدوین علی‌اکبر کسمایی

مظلومیت زن در طول تاریخ: نویسنده عبدالکریم بی‌آزار شیرازی

برابری زن و مرد در قصاص نوشته عباسعلی محمودی

پیشگیری و مبارزه با سرطان از دیدگاه اسلام: از دکتر سید محمدعلی سجادیه

سی مقاله: اثر استاد فخرالدین حجازی

پیشگیری و درمان نازائی: نوشته پروفسور جان‌شین. مترجمین: دکتر سعید، حبیب‌الله و ناصر ناظم

برداشتهای فقهی: نویسنده مرتضی رضوی نماینده مجلس شورای اسلامی

زبان برنامه‌نویسی بیسیک: ترجمه غلامرضا آذری

در دست انتشار:

تاریخ تحلیلی مطبوعات: نویسنده استاد محیط طباطبائی

بامید شکوفاتر شدن فرهنگ و معارف متعالی اسلام در تمام ابعاد علمی و فنی انشاءالله تعالی.

مؤسسه انتشارات بعثت

فهرست

پیشگفتار

نخستین برخورد من با کامپیوتر به سال‌های میانین دههٔ ۶۹—۱۹۶۰، هنگامی که من کار خود را همچون برنامه‌نویسی تازه کار آغاز کردم، باز می‌گردد. دستگاهی که با آن کار می‌کردم کامپیوتر بازرگانی نسبتاً بزرگی بود که از آن برای کارهای لیست گیری، همچون تهیهٔ صورت‌حساب‌ها، دفترهای بازرگانی، لیست حقوق و جز آن استفاده می‌شد. شرکتی که مرا به استخدام در آورده بود، بر روی این کامپیوتر، چه از نظر مالی و چه در زمینهٔ کارمندان، سرمایه‌گذاری بزرگی انجام داده بود.

اکنون که نزدیک بیست سال از آن زمان گذشته است، من «کامپیوتر خانگی*» کوچکی دارم که تقریباً در همهٔ زمینه‌ها از کامپیوتر پیشین بهتر است. این دستگاه دارای یک «سیستم عامل*» دیسکی جامع، یک انبارهٔ دیسکی* بزرگ، ٦٤ ک حافظه*، گزینه‌ای از «درگاههای»* ورودی و خروجی، یک «دستگاه نمایش*» تلویزیونی و «نرم افزاری»* گسترده (که زبان‌های برنامه‌نویسی «بیسیک*» و «کوبول*» را نیز در بردارد) می‌باشد.
در مقابل، کامپیوتری که من کارم را با آن آغاز کردم:

- سیستم عامل نداشت.
- انبارهٔ دیسکی نداشت.
- ٤ ک حافظه داشت.
- «ورودی*» آن تنها «کارت منگنه*» شده و «خروجی*» آن لیست‌های چاپ شده بود.
- دستگاه نمایش تلویزیونی نداشت
- دارای نرم افزار نبود. .
- زبان برنامه‌نویسی نداشت.

---

* یادآور مترجم: **واژه‌ها و عبارت‌هایی که با ستاره مشخّص شده‌اند اصطلاحات کامپیوتری می‌باشند. این اصطلاحات را در پیوست شمارهٔ ٤، توضیح اصطلاحات، شرح داده‌ایم.** مراجعه به واژه‌نامه‌های پایان کتاب نیز می‌تواند سودمند باشد. م

پس، بی گمان، در این زمینه پیشرفت بسیاری روی داده است.

سال‌های پایانی دههٔ هفتاد، که آغاز همگانی شدن کامپیوتر است، همیشه در یادها خواهد ماند. اکنون کامپیوتر را می‌توان از فروشگاههای خیابانی به دست آورد، و با هزینه‌ای کمتر از یک شب‌نشینی خرید و به خانه آورد. امروزه کامپیوتر در مدرسه، اداره، خانه و در حقیقت در همه جا به کار می‌رود.

بعضی از کامپیوترها تنها به درد کارهای ویژه‌ای می‌خورند؛ دستگاههای بزرگی که از آن‌ها برای کارهای بازرگانی، کنترل نیروی کارخانه‌ها و روشن و خاموش کردن چراغ‌های راهنمایی استفاده می‌شود به کار مردم عادی نمی‌آیند؛ هر چند کامپیوترهای خانگی در شمار چنان دستگاههایی نیستند. این کامپیوترها را همچنین «کامپیوتر شخصی*» «ریز کامپیوتر*» (و حتی «ریز پردازنده*») نیز می‌نامند. اگر دقیقتر بگوئیم، ریز پردازنده تنها یک «تراشهٔ*» **(chip)** سیلیسیمی است که در راندن کامپیوترهای خانگی به کار می‌رود، و برای این کار به تعدادی تراشهٔ دیگر (همچون حافظه، پردازنده‌های ورودی و خروجی، گیرنده‌های صفحه کلید و جز آن) نیازمند است.

مرزی که ریز کامپیوترها را از کامپیوترهای کوچک جدا می‌سازد درست روشن نیست. هرچند به طور کلّی، کامپیوترهایی را که در خانه، یا کسب و کارهای کوچک، به کار می‌روند ریز کامپیوتر می‌نامند. برنامه‌نویسی چنین دستگاههایی نباید تازه‌کاران را بترساند. برای این که این گفته را بپذیرید، تنها کافی است که ببینید کودکان چگونه به آسانی این دستگاهها را به کار برده و برای آن‌ها برنامه می‌نویسند. هدف این کتاب آن است که به تازه‌کاران زبان اصلی کامپیوترهای خانگی — یعنی بیسیک — را بیاموزد. بیسیک ساده‌ترین زبان‌ها است و بیش از هر زبان دیگری در کامپیوترهای خانگی به کار می‌رود. جز درباره کاربردهایی که به سرعت بسیار نیاز دارند، کاری نیست که زبان‌های دیگر برنامه‌نویسی انجام دهند ولی بیسیک در آن ناتوان باشد.

آموختن بیسیک دشوار نیست، برنامه‌نویسی هم کار دشواری نیست. بهای دستگاههایی که از ریز پردازنده‌ها استفاده می‌کنند روزبه‌روز کاسته می‌شود. و همراه آن، هزینهٔ آموختن برنامه‌نویسی نیز کاهش می‌یابد. دور نیست روزی که در

هر خانه‌ای، در کنار تلویزیون و دستگاه پخش، یک کامپیوتر هم جای گرفته باشد. آن گاه این کتاب شما را یاری خواهد کرد تا بهتر از آن دستگاه بهره‌برداری کنید.

بخش نخست

آشنایی با کامپیوترهای خانگی

این کتاب برای کسانی نوشته شده است که می‌خواهند روش برنامه‌نویسی بیسیک را برای کامپیوترهای خانگی بیاموزند. با وجود محدودیت‌هایی که در بخش دوّم بیان خواهد شد، مطالب این کتاب چنان فراهم شده است که خوانندگان بتوانند برای هر نوع کامپیوتری برنامه بنویسند.

با صرف مقدار کمی وقت و شکیبائی برای آموختن مطالب و مثال‌های این کتاب، هر کس می‌تواند برنامه‌نویسی به زبان بیسیک را بیاموزد. در کار کامپیوتر چشم‌بندی وجود ندارد و هیچ راز پوشیده‌ای نیست که دستیابی بر آن باعث پیوستن به باشگاه برگزیدگان (استفاده کنندگان کامپیوتر) گردد. از بخش سوم به بعد، آن چه برای یادگیری زبان برنامه‌نویسی بیسیک لازم است چنان عرضه خواهد شد که هم تازه‌کاران، و هم کسانی که با زبان‌های برنامه‌نویسی دیگر آشنا هستند. در آموختن مطالب با دشواری چندانی روبه‌رو نباشند. مطالب این کتاب را می‌توان بدون استفاده از کامپیوتر، و تنها با کمک مداد و کاغذ آموخت. هرچند این روش نمی‌تواند جای کار عملی با کامپیوتر را بگیرد.

تقریباً در همهٔ کامپیوترهای کوچک، زبان بیسیک قابل اجرا است. چنین دستگاههایی هرچند هم کوچک باشند، باز برای اطمینان بخشیدن به تازه‌کاران بسیار سودمند هستند. کامپیوترها از خود اراده‌ای ندارند، آنها تنها کارهایی را انجام می‌دهند که از آن‌ها خواسته می‌شود. شما به عنوان یک برنامه‌نویس می‌توانید این دستگاه را کاملاً در کنترل خویش داشته باشید. اگر در مورد به کار بستن اندیشه‌های خویش نگرانی دارید می‌توانید با چند تا از برنامه‌های آسانی که در دو بخش پایانی کتاب آمده‌اند، کار خود را آغاز کنید. به هر حال، به زودی کامپیوتر نشان خواهد داد که نه ارباب، بلکه بندهٔ شما می‌باشد.

بسیارند کسانی که می‌خواهند با کامپیوتر کار کنند ولی ترس روبرو شدن با دستگاه پیچیده‌ای از قطعات الکترونی آن‌ها را از این کار باز می‌دارد. هرچند کامپیوتر از نظر داخلی دستگاهی واقعاً پیچیده است، ولی نصب و راه‌اندازی آن از یک دستگاه پخش معمولی هم آسانتر است. کامپیوترهای خانگی گاهی تنها از یک دستگاه یا جعبه‌ای تشکیل شده‌اند که دارای وسایل الکترونی لازم در داخل و یک صفحه کلید بر روی آن می‌باشند. برای استفاده از این دستگاه باید آن را به تلویزیون منزل متصّل کرد (به محلی که آنتن هوایی وصل می‌شود)، و سپس آن را به برق زد. با این دو اتصال ساده اگر دستگاه نقص نداشته باشد آمادهٔ کار می‌شود. برخی از این کامپیوترها پس از روشن شدن نام خود را در بالای صفحهٔ تلویزیون ظاهر می‌نمایند (مثلاً APPLE II). همهٔ آن‌ها دارای «مکان‌نما*» می‌باشند. مکان‌نما معمولاً خط تیرهٔ چشمک‌زنی است که نشان دهنده جای «نشانهٔ*» بعدی در روی «صفحهٔ نمایش*» می‌باشد. حال اگر نشانه‌ای ماشین شود، این نشانه در جای مکان‌نما پدیدار شده و مکان‌نما به طرف راست برای نشان دادن جای بعدی حرکت می‌کند. اگر مکان‌نما به پایان یک سطر برسد، یا دکمهٔ «بازگشت*» (carriage return) فشار داده شود، مکان‌نما به اوّل سطر بعدی منتقل می‌شود.

کامپیوتر صفحهٔ نمایش را به تعداد مشخصی سطر (مثلاً ۲۴ سطر) و هر سطر را به تعدادی نشانه (مثلاً ۸۰ نشانه) تقسیم می‌نماید. وقتی که مکان‌نما به آخرین نشانهٔ آخرین سطر برسد، چنانچه نشانهٔ دیگری ماشین شود، این کار باعث می‌شود که سطر بالایی حذف شده، و هر سطر جای سطر قبلی را بگیرد، در نتیجه در پایین صفحه جای یک سطر خالی می‌شود.

صفحه کلید کامپیوتر، کم و بیش شبیه ماشین‌های تحریر معمولی می‌باشد، هر چند تعدادی کلید اضافی دارد. کلیدی که پایان یک سطر و شروع سطر بعدی را نشان می‌دهد دکمهٔ «بازگشت*» (RETURN) نامیده می‌شود. برخی از کامپیوترها، درست مانند ماشین‌های تحریر، حروف کوچک و بزرگ را با هم دارند. در اینجا هم هر کلید برای ماشین کردن دو نشانه به کار می‌رود، معمولاً برای ماشین کردن نشانه‌های بالایی باید «دکمهٔ جابه‌جایی*» (SHIFT) را همزمان فشار داد.

دو تا از کلیدهای اضافی کامپیوتر، «دکمه‌های گریز*» (ESCAPE) و «بازنشانی.*» (RESET) می‌باشند. کلید گریز در دستگاههای مختلف کارهای مختلفی را برعهده دارد که برای اطلاع یافتن از کار آن باید به کتاب راهنمای هر دستگاه مراجعه کرد. دکمهٔ بازنشانی برای «شروع دوبارهٔ» کار می‌باشد. فشار دادن این کلید به معنی از میان بردن اطلاعاتی است که قبلاً به دستگاه داده شده است و درست مانند این است که آن را خاموش و روشن کنید. از این رو باید کلید بازنشانی را با احتیاطی مورد استفاده قرار داد.

نمایش مکان‌نما و دریافت فرمان‌ها از این رو شدنی است که کامپیوتر تحت کنترل یک «سیستم عامل*» operating system قرار دارد. سیستم عامل برنامه‌ای است که توسط شرکت سازندهٔ کامپیوتر تهیه شده و در حافظهٔ دستگاه برای همیشه قرار داده شده است. این قسمت حافظه تنها قابل خواندن است (ROM) و از این رو نمی‌توان چیزی در آن نوشت و آن را از

میان برد. روشن کردن دستگاه، یا فشار دادن دکمهٔ بازنشانی باعث می‌شود که سیستم عامل کار خود را شروع نماید. بر روی صفحهٔ تلویزیون جمله‌ای همراه با مکان‌نما پدیدار شده و دستگاه منتظر فرمان می‌ماند. شروع برنامهٔ جدید با فرمان «نو» مشخّص می‌شود:

NEW

بعد از آن که برنامه به کامپیوتر داده شد آن را می‌توان با فرمان «راندن» اجرا کرد:

RUN

اگر برنامه قبلاً روی نوار کاست ضبط شده باشد با فرمان «بار کردن» می‌توان آن را خواند:

LOAD

برعکس برای ضبط یک برنامه از حافظه کامپیوتر بر روی نوار کاست فرمان «نگهداری کردن» به کار می‌رود:

SAVE

برای این که برنامه‌ای را از سطر نخست بر روی صفحهٔ تلویزیون نشان دهیم باید از فرمان «لیست گیری» استفاده کنیم:

LIST

چنانچه بخواهیم سطر مشخصی را نشان دهیم باید شمارهٔ آن سطر را نیز اطلاع دهیم:

LIST 300

و برای چاپ برنامه از یک سطر تا سطر دیگر، از فرمان زیر استفاده می‌کنیم:

LIST 250,400

علاوه بر این فرمان‌ها، فرمان‌های دیگری نیز هستند که در هر کامپیوتر مورد استفاده قرار می‌گیرند. هر چند فرمان‌های یاد شده فعلاً برای شروع کار، و به کار بستن دستورهای برنامه‌نویسی بخش‌های بعدی کتاب، کافی می‌باشند.

بخش دوم

آشنایی با زبان بیسیک

پیش از آن که یک کامپیوتر، چه کوچک و چه بزرگ، کاری انجام دهد (حال این کار حل یک مسئله باشد، یا شرکت در یک بازی، یا رسم یک تصویر و یا ...) باید روش انجام آن کار را یاد بگیرد. هر یک از مراحل کار و محاسبه باید به ترتیب و به شیوه‌ای منظم توضیح داده شود. به مجموعهٔ دستورهایی که برای انجام یک کار معین به کامپیوتر داده شده است «برنامهٔ کامپیوتر»، یا به طور خلاصه «برنامه» گفته می‌شود. برنامه باید چنان نوشته شود که کامپیوتر موردنظر بتواند آن را بفهمد. به سخن دیگر، کامپیوتر باید زبان برنامه‌ای را که به او داده می‌شود بداند.

بیسیک یکی از زبان‌های برنامه‌نویسی است. شاید بتوان گفت که بیسیک ساده‌ترین و آسان‌ترین زبان برنامه‌نویسی می‌باشد. هرچند بسیاری از برنامه‌نویسان حرفه‌ای زبان‌های پسندیده‌تر و نیرومندتری را به کار می‌گیرند. ولی زبان بیسیک برای کاربردهای غیرحرفه‌ای کاملاً توانا بوده و برای برنامه‌نویسان تازه کار بسیار دلپسند می‌باشد. تنها ضعفی که دارندگان کامپیوترهای شخصی ممکن است در بیسیک پیدا نمایند ناتوانی آن در انجام کارهایی است که باید به سرعت انجام شوند (چنانکه بعد خواهیم گفت، بیسیک زبانی است «تفسیری*» (interpretive)، و از این رو، درمقایسه با کارهای کامپیوتری تا حدی کند می‌باشد).

تاریخچه زبان بیسیک

واژهٔ بیسیک که از حرف‌های نخستین واژه‌های عبارت (Beginners All-purpose Symbolic Instruction Code) یعنی «زبان برنامه‌نویسی همه کاره برای تازه کاران» ساخته شده است، نام زبان برنامه‌نویسی است که در سال ۱۹٦٤، در کالج دارموث، درایالت نیوهامپشایر آمریکا ساخته شد و هدف از ساختن آن همان چیزی بود که از نام آن فهمیده می‌شود. این زبان در نخستین سال‌های دههٔ ۱۹۷۰ برای «کار همزمانی*» (Time-sharing) در کامپیوتر برگزیده شد. منظور از کار همزمانی آن است که چندین «پایانه*» (Terminal) همزمان با یک کامپیوتر بسیار نیرومند مرکزی، از راه دور، سرگرم کار باشند. کار همزمانی در کاربردهای مهندسی و حل مسایل، بسیار مورد استفاده قرار می‌گیرد.

در سال‌های پایانی دههٔ ۱۹۷۰ بیسیک زندگی نوینی را آغاز کرد، زیرا در این هنگام سازندگان کامپیوترهای کوچک آن را مورد استفاده قرار دادند. رشد کامپیوترهای خانگی باعث شده است که امروزه شمار **برنامه‌نویسان غیرحرفه‌ای بیسیک**، از کسانی که کار آن‌ها برنامه‌نویسی است **بیشتر باشد**.

گسترش زبان بیسیک پیدایش گویش‌های گوناگونی را به دنبال داشته است. اگر چه کوچکترین تفاوت در اجراءِ یا جمله‌بندی باعث می‌شود برنامه‌ای را که برای یک دستگاه نوشته‌ایم نتوانیم بدون تغییر در دستگاه دیگر اجراءِ نمائیم، ولی خوشبختانه این تفاوت‌ها بیشتر در بخش‌های کم کاربرد یا بسیار پیشرفتهٔ این زبان به وجود آمده است (مثلاً در ترسیم نمودارهای تلویزیونی با کامپیوترهای خانگی).

این کتاب هستهٔ واقعی زبان بیسیک را که برای تمام کامپیوترها مشترک است در بر می‌گیرد. هر چند نمونه‌هایی از موارد غیرمشترک، که تنها روی دستگاههای خاصی قابل اجراءِ هستند، نیز آورده می‌شود. هم «مؤسسهٔ استاندارد ملی آمریکا» (ANSI X3J2) و هم «مرکز محاسبهٔ ملی

انگلستان») (تعیین بیسیک استاندارد؛ نوشتهٔ بول، فری‌من، و گارلند) کوشیده‌اند که این هستهٔ مرکزی را تعریف و استاندارد نمایند. جز در جایی که یادآوری شده باشد. زبان بیسیکی که در این کتاب آمده است بدون تغییر در همهٔ دستگاهها قابل اجراءِ می‌باشد.

برنامه‌نویسی

هر زبان برنامه‌نویسی ابزاری است که برنامه‌نویس را در تعیین راه‌حل مسئله خویش، به زبانی که برایش آسانتر است، توانا می‌سازد (به گواهی همگان زبان‌هایی که به انگلیسی همانندند، همچون بیسیک، «کوبول*»، «پی ال و ان*» و «پاسکال*» از این شمارند.) برای این که کامپیوتر بتواند برنامهٔ خاصی را اجراءِ نماید نخست باید آن را به زبانی که خود می‌فهمد ترجمه نماید. این زبان که آن را «زبان هدف*» می‌نامیم معمولاً زبان اساسی ماشین می‌باشد. نوشتن برنامه مستقیماً به «زبان ماشین*» شدنی است هرچند که کاری بسیار خسته کننده و وقت‌گیر می‌باشد. برنامهٔ زبان ماشین باید به صورت «شانزده شانزدهی*» یا در پایهٔ شانزده نوشته شود. در زیر نمونه‌ای از زبان ماشین آورده می‌شود:

```
0041   27A3   410F
BC16   9016   A1C2
FF00   1350   2298
```

از این رو است که نیاز به داشتن زبان برنامه‌نویسی که «مهربان‌تر» باشد به زودی پدیدار می‌شود!

ترجمه زبان برنامه‌نویسی به زبان ماشین کاری است که بر دوش برنامهٔ «مترجم*» یا «مفسّر*» قرار می‌گیرد. مترجم همه برنامه را یکباره به برنامهٔ هدف که به زبان ماشین است برگردانده و همراه آن اشتباهات برنامه را مشخص می‌نماید.

مفسّر

بیشتر کامپیوترهای کوچک برای ترجمهٔ زبان بیسیک از مفسّر

استفاده می‌کنند. مفسّر به هر عبارت برنامه که بربخورد، آن را به زبان ماشین برمی‌گرداند. خواننده باید هنگام کار با مفسّر دو نکتهٔ مهم زیر را در نظر داشته باشد:

۱ـ هر عبارت داخل «حلقه*» (در بخش چهارم حلقه را توضیح خواهیم داد) در هر تکرار دوباره مورد تفسیر قرار می‌گیرد.

۲ـ برنامه ممکن است دارای اشتباهات پوشیده باشد، زیرا هنوز برنامه مورد آزمایش قرار نگرفته است. نخستین نکته نشان می‌دهد که چرا زبان بیسیک برای کارهایی که باید به سرعت انجام شوند، مانند بازی‌های کامپیوتری، چندان شایسته نیست. زبان بیسیک به وسیلهٔ دستورهایی که در بخش ششم خواهیم داد، وسیلهٔ ارتباط با زبان ماشین را فراهم می‌آورد.

دومین نکته بر لزوم آزمایش کامل برنامه تأکید می‌کند. همهٔ برنامه‌ها را باید با داده‌های آزمایشی جامع آزمود تا مطمئن شد که هر بخشی از برنامه دست کم یک بار به اجراءِ درآمده باشد.

عبارت‌های بیسیک را می‌توان به دو روش مستقیم و غیرمستقیم مورد تفسیر قرار داد. در روش مستقیم هر عبارت پس از آن که به دستگاه داده شد، با فشردن کلید بازگشت بی‌درنگ تفسیر شده و به اجراءِ گذارده می‌شود:

```
PRINT 27 + 13
40
```

در این مثال عبارت PRINT 27 + 13 بی‌درنگ تفسیر شده و کامپیوتر آن را با افزودن عدد ۲۷ به ۱۳ و چاپ نتیجه، یعنی ۴۰، بر روی صفحهٔ تلویزیون، به اجراءِ می‌گذارد (در این مرحله در مورد جزئیات و ریخت این عبارت نگران نباشید).

شمارهٔ سطرها

در برنامه‌نویسی بیسیک آن چه بیشتر به کار می‌رود روش غیرمستقیم است که در آن به هر سطری شماره‌ای داده می‌شود:

```
10 PRINT 27 + 13
20 PRINT "ANSWER"
30 GO TO 10
```

(همیشه در پایان هر سطر دکمهٔ بازگشت را فشار دهید).

در این روش هر عبارت پس از این که به کامپیوتر داده شد اجراءِ نمی‌شود، زیرا برنامه‌نویس هنوز نوشتن برنامه را دنبال می‌کند تا در پایان کار آن را به اجراءِ بگذارد.

پیش از آغاز کار باید به دستگاه اطلاع دهیم که می‌خواهیم برنامهٔ جدیدی بنویسیم. برای این کار به‌سادگی فرمان «نو» را ماشین می‌کنیم:

NEW

با این فرمان عبارت‌های برنامهٔ قبلی از حافظه ماشین پاک شده و از اشتباههایی که ممکن است بعداً پیش بیاید جلوگیری می‌شود. برنامه‌نویس باید سپس به ماشین کردن عبارت‌های برنامهٔ خویش، که به هر کدام شماره‌ای داده است، بپردازد. معمولاً عبارت‌ها را به ترتیب شماره سطر به کامپیوتر می‌دهند، هرچند از آن جا که دستگاه خود عبارت‌ها را به ترتیب شمارهٔ سطر در حافظه قرار می‌دهد، این کار لزومی ندارد:

10 عبارت الف

30 عبارت ج

20 عبارت ب

در هنگام اجراءِ برنامه به ترتیب زیر انجام خواهد شد:

عبارت الف

عبارت ب

عبارت ج

این ویژگی، برنامه‌نویس را توانا می‌سازد که عبارت‌های فراموش شده را بعدها به برنامه بیافزاید. در این مرحله شدیداً توصیه می‌کنیم که در

شماره‌گذاری سطرها از ضریب‌های عدد ۱۰ (یعنی ۱۰، ۲۰، ۳۰...) استفاده نمائید، کاری که ما در سرتاسر این کتاب انجام داده‌ایم. چنانچه یک شماره به دو عبارت داده شود، عبارت دومی به جای عبارت اوّلی قرار می‌گیرد: این روش برای اصلاح برنامه به کار می‌رود:

```
10 INPUT A
20 FOR J = 2 TO A
30 NEXT A
30 PRINT K(A)
40 NEXT A
20 FOR J = 1 TO A
15 PRINT "HEADING"
```

(در این مرحله نگران معنی این عبارت‌ها نباشید.)
به هنگام اجرا، این برنامه به شکل زیر مورد تفسیر قرار خواهد گرفت:

```
10 INPUT A
15 PRINT "HEADING"
20 FOR J = 1 TO A
30 PRINT K(A)
40 NEXT A
```

به یاد داشته باشد که در هر مرحله از نوشتن برنامه، می‌توانید با فرمان «لیست گیری» از کامپیوتر بخواهید که لیست درست برنامه را برای شما چاپ نماید:

LIST

با تغییر فرمان بالا می‌توان هر بخشی از برنامه را که بخواهید چاپ نمائید:

LIST 10,50

(برای چاپ عبارت‌هایی که شمارهٔ سطر آن‌ها میان ۱۰ تا ۵۰ می‌باشد.)

برای اجرای برنامه فرمان «راندن» را به کار می‌بریم:

RUN

این فرمان را به تفصیل بیشتر در بخش چهارم شرح خواهیم داد.

چنانچه بخواهیم دسته‌ای از عبارت‌ها را بارها مورد استفاده قرار دهیم می‌توانیم آن‌ها را روی دیسک یا نوار نگهداری نمائیم (بخش پنجم را ببینید). از این راه می‌توان برنامه‌های پیچیده را گام‌به‌گام به ماشین داد و هر مرحله را جداگانه آزمایش کرد.

بخش سوم

نخستین گام‌های برنامه‌نویسی

برنامهٔ شما برای هر کاری که نوشته شده باشد دارای خروجیی است که بر روی صفحهٔ تلویزیون نمایان می‌شود. این خروجی ممکن است نتیجه‌ای باشد که از یک محاسبه به دست آمده است و یا عبارتی که برای منظوری خاص فراهم شده است. همهٔ این خروجی های به وسیلهٔ دستور* چند کارهٔ «چاپ» (PRINT) انجام می‌شود. شاید شگفت‌آور باشد که نخست دستوری را شرح دهیم که بیشتر در پایان برنامه به کار می‌رود. ولی این دستور از آن جا که مستقیماً خروجی را تولید می‌نماید بیش از هر دستور دیگر مورد توجه قرار می‌گیرد.

از دستور چاپ برای فرستادن پیغام و نشان دادن آن بر روی صفحه نیز استفاده می‌شود. در ساده‌ترین شکل خویش این دستور از واژهٔ PRINT و پیغام موردنظر که در داخل علامت نقل‌قول قرار گرفته است، ساخته می‌شود:

```
PRINT "MY NAME IS JEFF"
MY NAME IS JEFF
```

توجه داشته باشید که متن پیام پس از واژهٔ PRINT آمده است؛ زیرا این یک دستور مستقیم می‌باشد (بخش دوم را ببینید). این دستور را می‌توانید به آسانی در کامپیوتر خویش بیازمایید؛ ولی فراموش نکنید که کلید بازگشت (RETURN) را در پایان عبارت فشار دهید.

در تمام مثال‌هایی که بعد از این خواهیم آورد برای وضوح بیشتر از روش غیرمستقیم استفاده خواهیم کرد یعنی هر سطری را شماره‌ای خواهیم داد:

```
100 PRINT "MY NAME IS JEFF"
```

برای اجرای این عبارت باید فرمان «راندن» (RUN) را به کار بریم، این فرمان به تفصیل در بخش چهارم شرح داده خواهد شد.

ثابت‌ها

دستور چاپ را می‌توان برای نشان دادن هر ترکیبی از حروف، اعداد و نشانه‌های دیگر به کار برد. با این دستور هر چیزی که در داخل علامت نقل‌قول قرار گیرد، جز خود آن علامت، چاپ خواهد شد:

```
100 PRINT "SAMPLE PRINT PROGRAM BY JM"
120 PRINT "SOME NUMBERS : - 12345"
140 PRINT "OTHER CHARS :-?()!./"
```

آن چه که در داخل علامت نقل‌قول می گیرد «ثابت*» نامیده می‌شود. ارزش ثابت‌ها در طول اجرای برنامه تغییر نمی‌کند. در دستگاههای گوناگون ثابت‌ها را به نام‌های دیگری همچون «رشته*»، «رشتهٔ حرفی*» «رشتهٔ حروف و عددی*» و جز آن می‌نامند. ثابت را می‌توان به شکل اعداد نیز نشان داد که در این صورت به آن «ثابت عددی» گفته می‌شود:

```
200 PRINT 27, 34.6
```

در دستور بالا نخستین ثابت عددی، یک عدد صحیح می‌باشد. این ها نمونه‌های دیگری از ثابت‌های عددی می‌باشند.

39,46,2164392.

توجه کنید که برای جدا کردن عددها از یکدیگر از ویرگول استفاده می‌شود و بین ارقام هر عدد نباید فاصله‌ای باشد. از این رو ۳۹۲ و ۱٦٤ و ۲ سه عدد به شمار می‌آیند نه یک عدد.

دومین ثابت عبارت بالا عددی است اعشاری که دارای ممیّز می‌باشد:

```
220 PRINT 214.61,3.1419,6.0
```

در سطر ۲۲۰ آخرین ثابت، یعنی ۶/۰ هم عددی است اعشاری، زیرا دارای ممیز می‌باشد، هر چند مقدار آن با عدد صحیح ۶ برابر می‌باشد.

ثابت‌های عددی منفی را، چه صحیح باشند و چه اعشاری، با علامت منفی در طرف چپ نشان می‌دهیم:

```
240 PRINT - 46,-0.39,16.1
```

بزرگترین عددی که بیسیک می‌تواند بپذیرد، و همچنین تعداد ارقام اعشاری که از محاسبه به دست می‌آید، به نوع دستگاه بستگی دارد.

ارزش ثابت‌ها، چنان که از نامشان برمی‌آید در طول اجرای برنامه تغییر نمی‌کند. ولی متغیّرها (Variables) در ضمن اجرای برنامه می‌توانند مقدارهای مختلفی را بپذیرند. همچون ثابت‌ها، متغیّرها نیز می‌توانند دارای مقدارهای عددی یا رشته‌ای (حرف و عددی) باشند.

متغیّرها

متغیّرهای عددی را که می‌توانند ارزش‌های عددی صحیح یا اعشاری بپذیرند، به سادگی به وسیلهٔ یک حرف نشان می‌دهیم. از این رو هر برنامهٔ بیسیک می‌تواند حداکثر بیست و شش تا، یعنی به تعداد حرف‌های خط انگلیسی، ثابت عددی داشته باشد.

برای این که به متغیّری مقداری را واگذار نماییم دستور «واگذاری*» (LET) را به کار می‌بریم:

```
100 LET C = 27
```

این عبارت می‌گوید که: «مقدار ۲۷ را به متغیّر C واگذار کن». با این دستور مقدار ۲۷ در متغیّر قرار گرفته و مقدار قبلی آن از میان می‌رود. مقدار واگذار شده ممکن است از یک محاسبه به دست آمده باشد:

```
110 LET D = 27 + 13 + 6
```

در این مثال مقدار D برابر ۴۶ خواهد شد.

معمولاً در محاسبه از متغیّرهای دیگر نیز استفاده می‌شود:

```
150 LET A = B + 1
160 LET A = 2 *A
```

در سطر ۱۵۰ مقدار B با ۱ جمع شده و نتیجه به A واگذار می‌شود. در سطر ۱۶۰ مقدار A دو برابر می‌شود.

محاسبه

عبارتی را که در طرف راست علامت مساوی قرار می‌گیرد می‌توان با استفاده از علامت‌های محاسبه‌ای بسیار پیچیده‌تر کرد. علامت‌های زیر قابل قبول می‌باشند:

↑ توان

* ضرب

/ تقسیم

+ جمع

− تفریق

زبان بیسیک علامت‌های بالا را به همان ترتیبی که آورده‌ایم به کار می‌برد (یعنی نخست علامت توان را، سپس علامت‌های ضرب و تقسیم را، و در پایان علامت‌های جمع و تفریق را). اهمیّت این موضوع را می‌توان با عبارت زیر نشان داد:

```
130 LET A = B/C/D
```

آیا منظور محاسبهٔ D / (A/C) یا A/ (C/D) می‌باشد؟ نتیجه هر عبارت با دیگری کاملاً متفاوت است (می‌توانید با اعداد ۸ و ٤ و ۲ آزمایش کنید).

هر جا که در مورد ترتیب انجام محاسبات شک داشته باشید می‌توانید

با افزودن پرانتزهای اضافی نگرانی خود را از میان بردارید. از پرانتز همچنین می‌توان برای وضوح بیشتر، حتی در جایی که مورد نیاز نباشند، استفاده کرد.
برای نمونه در عبارت زیر سطح جانبی یک استوانه را محاسبه کرده‌ایم:

```
270 A = (3.14*(D ↑ 2/4)) + (3.14*D*H)
```

توجه کنید که واژهٔ LET را حذف کرده‌ایم زیرا در بیشتر گونه‌های* بیسیک می‌توان این کار را کرد (و در این کتاب بعد از این همین روش را به کار خواهیم بست). مثال‌های بیشتری برای نشان دادن دستور واگذاری در بقیّه این بخش خواهیم آورد.

عبارت واگذاری مقدار جدیدی را به متغیّرها واگذار می‌نماید و با این کار مقدار قبلی آن را نابود می‌کند. می‌توان عبارت‌های پیچیده را به چند عبارت ساده‌تر تبدیل کرد. در این صورت باید از متغیرهای بیشتری استفاده کرد:

```
140 P = 3.14 *D ↑ 2/4
150 Q = 3.14*D*H
160 S = P + Q
170 R = S/C
```

یا از یک متغیّر (مثلاً ازP) چند بار استفاده نمود:

```
140 P = 3.14*D↑2/4
160 P = P + 3.14*D*H
170 R = P/C
```

نتیجه‌ای که از دو مثال بالا به دست می‌آید یکسان است. برخی از گونه‌های بیسیک در آغاز کار مقدار همه متغیرها را صفر می‌کنند، ولی بعضی این کار را نمی‌کنند. چنانچه از طرز کار بیسیک خود اطمینان ندارید، می‌توانید خود در آغاز کار به متغیّرها مقدار صفر را واگذار نمائید:

```
10 A = 0
20 B = 0
30 C = 0
```

در برخی از گونه های بیسیک می‌توان عبارت زیر را به کار برد:

```
10 A = B = C = 0
```

در بعضی از گونه های زبان بیسیک می‌توان نام متغیر را از یک حرف و یک عدد انتخاب کرد؛ مثلاً: A1,A2, – Z1, Z2, – Z9

بیست و شش متغیّر حرفی*، یعنی Z$...B$, A$ ، برای نگهداری مقدارهای حرفی به کار می‌روند (به همان شیوه‌ای که در ثابت های حرفی دیدیم). برای واگذاری مقدارهای حرفی نیز از علامت مساوی استفاده می‌کنیم. طول رشتهٔ حرفی به نوع دستگاه بستگی دارد:

```
240 N$ = "J MAYNARD "
250 A$ = "AUTHOR"
400 PRINT N$ + A$
```

عبارت ٤٠٠ نتیجهٔ زیر را بر روی صفحه نمایش نمایان خواهد ساخت:

```
J MAYNARD AUTHOR
```

علامت جمع هنگامی که با متغیرهای حرفی به کار رود آن‌ها را به یکدیگر می‌چسباند.

متغیرهای حرفی نیز، همچون متغیرهای عددی، در برخی از دستگاهها در آغاز کار پاک می‌شوند و در بعضی پاک نمی‌شوند. در این مورد هم چنانچه شک دارید می‌توانید در آغاز برنامه با استفاده از دستور واگذاری آن‌ها را پاک نمائید.

با دستور چاپ می‌توان آمیخته‌ای از ثابت ها و متغیّرها را بر روی صفحه نشان داد. مثال:

```
210 X$ = " EQUALS "
220 PRINT "THE SQUARE OF "; B;X$ ; B*B
```

این دستور در بردارندهٔ یک ثابت حرفی، یک متغیّر عددی، یک متغیّر حرفی و یک عبارت محاسبه‌ای می‌باشد. اگر مقدار B برابر ۲ باشد، نتیجه بر روی صفحهٔ نمایش به صورت زیر خواهد بود:

```
THE SQUARE OF 2 EQUALS 4
```

متغیّری که در دستور چاپ جای می‌گیرد، مقداری را که در آن زمان دارد، بر روی صفحه نشان می‌دهد؛ عبارت‌های محاسبه‌ای نیز محاسبه می‌شوند و نتیجهٔ آن‌ها پدیدار می‌شود. این عبارت‌ها می‌توانند به صورت پیچیده‌تری نیز در دستور قرار گیرند:

```
440 PRINT "AREA IS "; 3.14 *D↑2/4
```

علامت نقطه ویرگول را هنگامی به کار می‌بریم که بخواهیم میان مقدارهای چاپ شده فاصله‌ای گذاشته نشود. اگر این علامت را در پایان سطر به کار ببریم، دستور چاپ بعدی، آن چه را که بخواهد نمایان سازد، به دنبالهٔ همان سطر چاپ شده، اضافه می‌کند:

```
550 PRINT  "NAME:  ";
560 PRINT N$
```

اگر ارزش N$ برابر "SMITH" باشد، نتیجه چاپ به صورت زیر خواهد بود:

```
NAME:  SMITH
```

اگر دستور چاپ را به تنهایی به کار ببریم یک سطر نا نوشته می‌ماند:

```
600 PRINT
```

از این روش برای خواناتر بودن پاسخ‌های چند سطری استفاده می‌کنیم.

نام متغیّر

در گونه های جدید بیسیک برخلاف آنچه تاکنون دیدیم، که هر متغیّر با یک حرف نشان داده می‌شد، می‌توان برای متغیرها نام‌های بامعنی انتخاب کرد. برای نمونه، برنامه‌ای که برای محاسبهٔ دستمزد به کار می‌رود می‌تواند دارای عبارتی به شکل زیر باشد:

```
220 NET = GROSS − (TAX + DEDUC)
```

در یک برنامهٔ بازی که برای بهتر نشان دادن حرکت توپ در برابر دیوار از رنگ استفاده می‌شود نام‌های زیر کاربرد متغیّرها را بسیار آسانتر می‌کنند.

```
230 BALL = COLOUR1
240 WALL = COLOUR2
```

برای متغیرهای حرفی نیز می‌توانیم از نام‌های معنی دار، چنان که در مثال زیر می‌بینیم، استفاده نمائیم. نخست به بخشی از یک برنامهٔ چاپ «صورت حساب» توجه نمائید:

```
410 ON ACCOUNT GO TO 440,460,480
420 TERM$ = "CASH"
430 GO TO 490
440 TERM$ = "C.O.D."
450 GO TO 490
460 TERM$ = "CREDIT CARD"
470 GO TO 490
480 TERM$ = "ACCOUNT"
490 . . .
```

متغیّر نماینده (یا شاخص) ACCOUNT ، روش پرداخت را که باید بر روی صورت‌حساب مشتری چاپ شود، نشان می‌دهد (این روش در متغیّر حرفی TERM$ نگهداری می‌شود).

دوّم، بخشی از یک برنامهٔ بازی را می‌آوریم که همان روش برای خواناتر بودن در آن به کار رفته است. نام «دشمن» در بازی جنگ ستارگان، در متغیر نماینده‌ای به نام ENEMY (=دشمن) نگهداری شده است. نامی که باید به وسیله دستور زیر چاپ شود به متغیّر حرفیی به نام OPPONENT$ (=هم نبرد) واگذار می‌شود:

```
260 PRINT "YOU HAVE CONQUERED THE
    OPPONENT$
```

دستور واگذاری بعد از آزمایش مقدار ENEMY انجام می‌شود:

```
220 IF ENEMY = 1 THEN 250
230 OPPONENT$ = "ZARGON WARRIORS"
240 GO TO 260
250 OPPONENT$ = "PHOXYLENIC SHIP"
```

زبان بیسیک دارای تعدادی «واژهٔ اندوخته *» (Reserved) می‌باشد. این واژه‌ها در زبان بیسیک دارای معنی ویژه‌ای می‌باشند. برای نمونه در زیر چند واژهٔ اندوخته را می‌آوریم:

LET PRINT FOR NEXT RUN

به طور کلی هر واژه‌ای که برای دادن فرمان یا دستوری در بیسیک به کار رود در شمار واژه‌های اندوخته می‌باشد. به یاد داشته باشید که از واژه‌های اندوخته نمی‌توان به عنوان نام متغیّر استفاده کرد، و حتی نمی‌توان آن‌ها را در داخل واژه‌های دیگر به کار برد. از این رو نام‌های زیر را نمی‌توان

به متغیرها داد:

LETTUCE
FORTUNE
RUNNING
SPRINT

برای این که برنامه‌های نمونه در همهٔ کامپیوترها قابل اجراءِ باشند، در مثال‌هایی که بعد از این خواهیم آورد، برای متغیرها نام‌های ساده‌ای انتخاب خواهیم کرد (A Z A$ Z$).

جدول‌بندی

اگر در دستور چاپ، نام‌ها و ثابت‌ها را از یکدیگر با ویرگول جدا کنیم، محتویات آن‌ها با فاصله‌های معینی از یکدیگر چاپ خواهند شد. اگر صفحهٔ نمایش بتواند در طر سطر ۸۰ نشانه را نشان دهد در این صورت هر سطر به پنج بخش، که هر کدام گنجایش ۱۵ نشانه را دارد، تقسیم می‌شود. هر ویرگول در دستور چاپ باعث می‌شود که نمایش محتویات نام یا ثابت بعدی از اوّل بخش بعدی شروع شود. چنانچه صفحهٔ نمایش دارای سطرهای ٤۰ نشانه‌ای باشد جدول‌بندی کاربرد کمتری دارد؛ توصیه می‌شود در این مورد به کتاب راهنمای دستگاه مراجعه شود. البته می‌توان از دستور جدول‌بندی (PRINT TAB) برای تعیین جای چاپ استفاده کرد. برای نمونه عبارت زیر باعث می‌شود که مقدار در جای بیست و یکم سطر چاپ شود:

640 PRINT TAB (20); A

عدد داخل پرانتز جلوی TAB می‌تواند از صفر تا ۳۹ تغییر نماید. دستور بالا که در همهٔ گونه‌های بیسیک وجود دارد، جای چاپ را فقط در روی سطر موجود تعیین می‌نماید. گاهی لازم است که شماره سطر و جای چاپ را خود تعیین نمائیم. معمولاً کامپیوترهای کوچک بر صفحهٔ تلویزیون ۱٦ یا ۲٤ سطر ۳۲، ٤۰ یا ٦٤ نشانه‌ای را نمایش می‌دهند. با استفاده از دستور PRINT AT می‌توان ستون خاصّی را در سطر خاصّی تعیین کرد. (توجه

داشته باشید که شماره بالاترین سطر صفر و شماره اولین ستون دست چپ نیز صفر می‌باشد).

دستور زیر در وسط صفحه‌ای که دارای ۲٤ سطر ۳۲ نشانه‌ای باشد یک ستاره چاپ می‌کند:

```
PRINT AT 11,15;"*"
```

می‌توان برای تعیین سطر و ستون از متغیر استفاده کرد.

```
PRINT AT L,C*2;"+"
```

با این دستور می‌توان هر نقطه‌ای از صفحه را برای چاپ انتخاب کرد، حتی اگر قبلاً در آن محّل چیزی چاپ شده باشد. البته با این کار جای نشانهٔ چاپ شدهٔ قبلی را، نشانهٔ جدید می‌گیرد. از این رو باید در استفاده از این دستور دقت لازم را به کار برد.

پاک کردن صفحهٔ تلویزیون

برای پاک کردن صفحه تلویزیون و همچنین به خانه آوردن مکان‌نما (گوشهٔ بالا دست چپ)، دستورهای خاصّی وجود دارد. این دستورها در دستگاههای مختلف ممکن است متفاوت باشند. در کامپیوتر اَپِل (Apple) چنین می‌باشند:

```
20 CLEAR
30 HOME
```

دستورهای جدول‌بندی صفحه و تغییر مکان‌نما برای شسته و رفته نشان دادن خروجی برنامه بسیار مفید می‌باشند. مثلاً برنامه‌ای که برای تهیه اطلاعات آماری نوشته شده است، اگر اطلاعات را روشن و فشرده نمایش دهد بسیار پذیرفته خواهد بود.

عبارت «داده‌ها» و دستور «خواندن»

هر برنامه دارای ثابت‌ها و «داده‌ها*»ی خامی می‌باشد که آن‌ها را در محاسبه به کار می‌برد. چنان که دیدیم برای پر کردن متغیّرها از دستور

واگذاری استفاده می‌کنیم:

```
100 A = 27.0
120 B = 76
140 N$ = "ACCESS NO"
```

هر برنامه‌ای به تعدادی ثابت نیاز دارد هر چند می‌توان به جای آن‌ها عبارت «داده‌ها» و دستور «خواندن» (READ) را برای پر کردن متغیّرها در هنگام اجرای برنامه، به کار برد.

عبارت داده‌ها (DATA) زنجیره‌ای از ثابت‌های عددی یا حرفی را در برمی‌گیرد:

```
110 DATA 18, 419.6, "TOTAL", "NEXT"
```

به وسیلهٔ دستور خواندن می‌توان بر مقدارهای زنجیرهٔ داده‌ها دست یافت. دستور زیر باعث می‌شود که مقدارهای ۱۸ و ۴۱۹/۶ به ترتیب به متغیّرهای E و F واگذار شوند:

```
240 READ E,F
```

در دستور خواندن می‌توان چندین متغیّر به کار برد، به شرط اینکه نخست در عبارت داده‌ها به اندازهٔ کافی ثابت وجود داشته باشد؛ و دوم، نوع متغیرها با نوع داده‌های مربوط به آن‌ها جور باشد. از این رو دستور خواندن بعدی، که باید ثابت‌های "TOTAL" و "NEXT" را بخواند، می‌تواند به صورت زیر باشد:

```
250 READ T$, N$
```

چنانچه دستور خواندن را به کار بریم ولی در عبارت داده‌ها دیگر ثابتی وجود نداشته باشد، این کار باعث پایان یافتن برنامه و فرستادن پیغام نادرست بودن دستور خواهد شد:

```
OUT OF DATA IN 500
```

در این جا عدد ۵۰۰ نشان دهندهٔ شمارهٔ سطر دستور خواندن می‌باشد.

فرض کنید که در برنامه، یک عبارت دیگر به صورت زیر وجود داشته

باشد:

```
120 DATA 49, 216.8, "TOTAL", "LAST"
```

اجرای دوباره سطر ۲۴۰ متغیرهای E و F را به ترتیب با مقدارهای ۴۹ و ۲۱۶/۸ پر خواهد کرد. همچنین اجرای دوباره دستور ۲۵۰ ثابت‌های "TOTAL" و "LAST" را به متغیرهای T$ و N$ واگذار خواهد کرد. توجه نمائید که هرچند مقدار T$ در هر دو اجرا یکسان است، ولی برای این که دستور ۲۵۰ در بار دوم هم درست اجراءِ شود، لازم است که در عبارت ۱۲۰ ثابت "TOTAL" دوباره معرفی شود.

بیسیک دارای «نشانگر*»ی است که نشان دهندهٔ ثابتی که باید با دستور خواندن بعدی برداشته شود، می‌باشد. هر متغیر دستور خواندن یکی به مقدار این نشانگر می‌افزاید. با دستور زیر می‌توان مقدار «نشانگر» را برابر یک قرار داد. با این کار نخستین ثابت از نخستین عبارت داده‌ها، نشان داده می‌شود.

```
200 RESTORE
```

خلاصه

برنامه‌های زیر نشان دهندهٔ ویژگی‌هایی که در این بخش دیدیم می‌باشند.

نمونه ۱

این برنامه برای برگرداندن فوت به مترنوشته شده است. مقدارهایی که باید برگردانده شوند در عبارت داده‌های شماره ۱۰۰ نشان داده شده‌اند. هر مقدار نخست بر ۳ بخش شده تا یارد به دست آید، سپس مقدار به دست آمده در عدد ۰/۹۱۴۴ ضرب شده تا مقدار برحسب متر به دست آید:

```
100 DATA 12,6,4,3
110 PRINT
120 READ F
130 Y = F/3
140 M = Y*0.9144
150 PRINT F; " FEET EQUALS "; M; " METRES "
160 GO TO 120
170 END
```

دستور شمارهٔ ۱۶۰ کنترل برنامه را دوباره به سطر شماره ۱۲۰ برمی‌گرداند، این کار باعث می‌شود که دستور خواندن دوباره انجام شود. (دستور «جهش*» (GO TO) را در بخش ۴ به تفضیل شرح خواهیم داد). این برنامه تا هنگامی که در عبارت داده‌ها مقدای وجود داشته باشد اجراءِ خواهد شد. هنگامی که ثابت‌های آن عبارت به پایان برسند برنامه با پیغام OUT OF DATA IN LINE 120 کار خود را پایان خواهد داد (برای پایان بخشیدن به برنامه روش‌های بهتری هم وجود دارد که بعد خواهیم دید). دستور پایان END که در سطر شمارهٔ ۱۷۰ آمده است نشان‌دهندهٔ پایان پیکرهٔ برنامه می‌باشد.

اگر به برنامه فرمان راندن (RUN) داده شود پاسخ‌های زیر پدیدار خواهند شد:

```
RUN
12 FEET EQUALS 3.66 METRES
6 FEET EQUALS 1.83 METRES
4 FEET EQUALS 1.22 METRES
3 FEET EQUALS 0.91 METRES
OUT OF DATA IN LINE 120
```

می‌توان با تغییر سطر شماره ۱۰۰ مقدارهای دیگر را نیز محاسبه کرد.

```
100 DATA 18,16,216,92,200
```

برای این کار سطر بالا را ماشین کنید و سپس فرمانِ «راندن» را به دستگاه بدهید.

نمونهٔ ۲

این برنامه هزینهٔ مأموریت کارمندان یک شرکت را محاسبه می‌کند. هزینهٔ هر مأموریّت عبارت است از حقوق، حق مأموریت و هزینه‌های دیگر کارمند که بر تعداد مأموریت‌های او تقسیم می‌شود. داده‌های مربوط به هر کارمند برای روشن‌تر شدن برنامه، در یک عبارت جداگانه آمده است. خروجی برنامه بعد از فرمانِ «راندن» نشان داده شده است.

```
100 DATA "SMITH", 4000,3000,1000,163
110 DATA "JONES", 4100,2860,1200,170
120 DATA "BROWN", 4260,3110,1017,166
130 PRINT
140 READ N$,S,C,E,Q
150 X = (S + C + E)/Q
160 PRINT N$;" COSTS £";X;" PER CALL"
170 GO TO 140
180 END
```

```
RUN

SMITH COSTS £49.08 PER CALL
JONES COSTS £48 PER CALL
BROWN COSTS £50.52 PER CALL
OUT OF DATA IN LINE 140
```

یادآوری: گفتیم که برای توان نشانهٔ ↑ به کار می‌رود. در بعضی از دستگاهها به جای این نشانه از دو ستاره استفاده می‌کنند:

```
120 C = A**2
```

برخی از دستگاههای ساده‌تر نمی‌توانند عمل توان را به صورت محاسبه‌ای انجام دهند، بلکه به جای آن از تابع استفاده می‌کنند (تابع ها را در بخش ۹ شرح خواهیم داد).

بخش چهارم

کنترل برنامه

در برنامه‌های پایانی بخش ۳ با دستور جهش (GO TO) آشنا شدیم. این نوع جهش را جهش غیرشرطی * می‌نامیم، زیرا هر وقت برنامه این دستور را انجام دهد، پس از آن بی درنگ کار را از سطری که شمارهٔ آن مشخص شده است ادامه خواهد داد:

```
100 GO TO 220
```

این دستور کنترل برنامه را بدون هیچ شرطی به سطر شماره ۲۰۰ منتقل می‌کند.

دقت نمائید که از دستور بالا به صورت زیرا ستفاده نکنید:

```
410 GO TO 410
```

زیرا این دستور کنترل را به خودش منتقل می‌نماید و این عمل بارها بدون توقف ادامه می‌یابد. برای خارج شدن از این حلقهٔ بی‌پایان باید از کلیدهای دستگاه استفاده کرد (مثلاً یکی از کلیدهای بازنشانی (RESET)، ایست STOP یا گریز (ESCAP) را فشار داد). این کار ممکن است باعث از دست دادن اطلاعات شود.

در برخی از گونه‌های زبان بیسیک می‌توان در دستور جهش شمارهٔ سطر را از محاسبه به دست آورد:

```
120 GO TO 2*A + 10
```

در هنگام اجراء، نتیجه محاسبه شماره سطر هدف را مشخص می‌نماید. البته این روش چندان مناسب نیست زیرا ممکن است نتیجه محاسبه عددی باشد که چنان شمارهٔ سطری در برنامه وجود نداشته باشد. روش‌های بهتری برای این کار وجود دارد که با استفاده از مقدار یک متغیر این عمل را انجام می‌دهد. این روش‌ها را بعداً توضیح خواهیم داد.

یکی از روش‌های جهش، که بسته به مقدار یک یا چند متغیر این عمل انجام می‌شود یا نمی‌شود، «جهش شرطی*» نام دارد. در این جهش در صورتی کنترل به یک شماره سطر منتقل می‌شود که شرایط موردنظر برقرار باشد. برای این جهش دستور «اگر— پس» (IF-THEN) را به کار می‌بریم.

در دستور «اگر— پس» هنگامی جهش انجام می‌شود که حالتی درست باشد (یا شرطی برقرار باشد). اگر حالت درست نباشد (یا شرط برقرار نباشد) دستور بعد از «اگر— پس» انجام خواهد شد.

نخست نمونهٔ ساده‌ای از دستور «اگر— پس» را به دست می‌دهیم. برنامه داده‌هایی را می‌خواند، مقدار صفر پایان داده‌ها را مشخص می‌کند، هنگامی که داده‌ای با مقدار صفر خوانده شد کنترل باید به بخش پایانی برنامه جهش نماید:

```
100 DATA 26.7,-139.14,863,2291.6,0
.
.
.
200 READ A
210 IF A = 0 THEN 900
220 B = A*2
900 PRINT "END OF RUN"
999 END
```

سطر شماره ۲۱۰ شرط A = 0 را بررسی می‌نماید. اگر این شرط درست باشد (یعنی A = 0 باشد) کنترل به سطر شماره ۹۰۰ جهش خواهد

کرد. اگر مقدار A نه صفر باشد (شرط درست نباشد)، برنامه دستور بعدی (سطر شماره ۲۲۰) را انجام خواهد داد.

با دستور اگر— پس می‌توان شرایط زیر را بررسی کرد:

شرط	دستورهای نمونه
برابر است با	100 IF A = B THEN 220 110 IF A = 7 THEN 700
بزرگتر است از	210 IF A>D THEN 160 220 IF B>0 THEN 160
کوچکتر است از	340 IF 0<D THEN 270 350 IF B<1 THEN 280 360 IF C<D THEN 270
بزرگتر یا مساوی است با	260 IF R> = 296 THEN 120 270 IF R> = S THEN 440
کوچکتر یا مساوی است با	300 IF A< = 0 THEN 550 310 IF A< = P THEN 640
نابرابر است با	480 IF A<>B THEN 190 490 IF A<>64 THEN 180

هر یک از مقدارهای مورد مقایسه می‌تواند یک عبارت محاسبه‌ای باشد:

```
140 IF A > (B + 7)* D THEN 220
150 IF R <> (A*B) + (C/D) THEN 230
160 IF (A + 7)*2 = B + (C*D)/7 THEN 460
```

دستور اگر— پس را می‌توان برای بررسی متغیرهای حرفی یا ثابت‌های حرفی نیز به کار برد. در این مورد باید تنها نشانه‌های = (برابر است با) و <> (نابرابر است با) را به کار برد.

```
470 IF N$ = "NEWNES" THEN 200
480 IF N$ = R$ THEN 500
620 IF A$ <> "END" THEN 100
```

(البته در برخی از بیسیک‌ها می‌توان نشانه‌های دیگر را نیز به کار برد.)

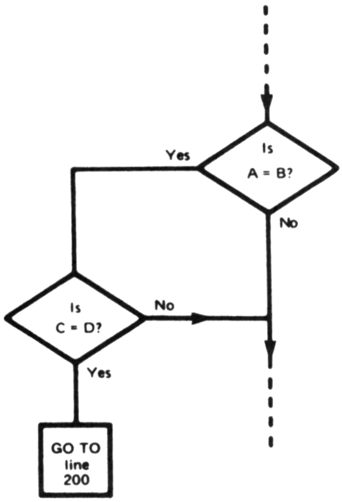
توجه داشته باشید که دو متغیر حرفی " YES" و "YES" با یکدیگر برابر نیستند زیرا اوّلی قبل از حرف Y یک «نشانهٔ تهی*» (SPACE) دارد و دومی ندارد. نشانهٔ تهی همچون نشانه‌های دیگر، مثلاً Y، مورد مقایسه قرار می‌گیرد. مقایسه متغیّرهای حرفی و عددی کاری است بی‌معنی و هرگز نباید چنین کاری را انجام داد.

در بعضی از گونه‌های بیسیک دستور اگر ـ پس را می‌توان به صورت چندگانه یا تودرتو انجام داد:

100 IF A = B THEN IF C = D THEN 200

(دستور بالا را گاهی به صورت IF A = B IF C = D THEN 200) هم می‌نویسند.) شرط دوّم (C = D) هنگامی بررسی می‌شود که شرط نخست (یعنی A = B) درست باشد. از این رو کنترل هنگامی به سطر ۲۰۰ جهش خواهد کرد که هر دو شرط درست باشد. دستور بالا را به وسیلهٔ «گردش‌نمای*» زیر نشان داده‌ایم (گردش‌نما نموداری است که روند اجرای برنامه را نمایش می‌دهد).

شکل ۱ـ۴ گردش‌نمای «اگر- پس» تودرتو



گردش‌نما وسیلهٔ بسیار سودمندی است که برنامه‌نویس می‌تواند از آن استفاده نماید. می‌توان گردش‌نما را برای روشنتر کردن مسئله‌ای که فهم آن دشوار است به کار برد. هرگونه تردیدی را که در مورد منطق برنامه وجود داشته باشد، باید با کشیدن گردش‌نما از میان برد، حتی اگر آن را با شتاب و سردستی فراهم آورده باشیم.

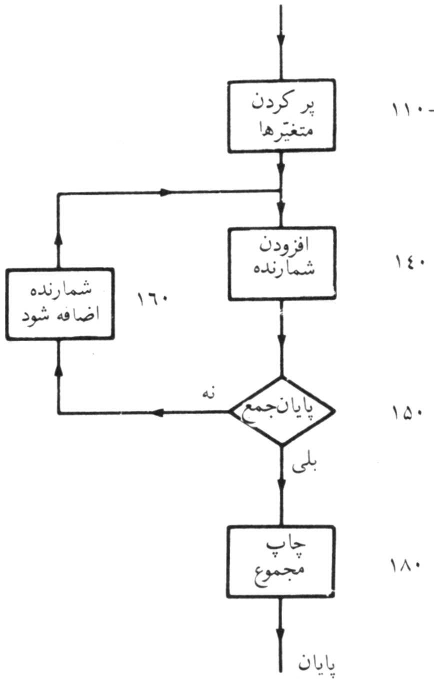
برنامه‌نویسان حرفه‌ای برای نشادن کارهای گوناگون از شکل‌های گوناگونی در گردش‌نما استفاده می‌کنند. هرچند برای نشان دادن بیشتر کارها تنها دو شکل را می‌توان به کار برد. این دو شکل نشانه‌های سنجش (یعنی لوزی) و کنش (یعنی مستطیل) می‌باشد. (در پیوست ۳ــ «نمادهای گردش‌نمایی»ــ نشانه‌های بیشتری را نشان داده‌ایم.)

هر نشانه سنجش دارای یک راه ورودی (معمولاً از گوشهٔ بالا) و دو یا سه راه خروجی می‌باشد (بعضی‌ها بهتر می‌دانند که نشانهٔ سنجش تنها دو

راه خروجی داشته باشد). راهی را که برنامه در هنگام خروج از نشانهٔ سنجش برمی‌گزیند به نتیجهٔ سنجش یا درست و نادرست بودن شرط ها بستگی دارد. در برنامه به جای نشانهٔ سنجش معمولاً دستور اگرــ پس را به کار برمی‌برند ، نشانهٔ کنش تنها یک راه ورودی و یک راه خروجی دارد و بسته به جای آن در گردش‌نما، می‌توان این راه‌ها را به هر کدام از اضلاع مستطیل که بهتر است مربوط ساخت، هرچند باید جهت گردش را به روشنی با به کار بردن پیکان نشان داد (شکل ۱ــ ۴). دستورهای داخل نشانهٔ کنش هنگامی که برنامه به آن وارد می‌شود حتماً انجام می‌شوند.

در مثال بعدی دستور اگرــ پس، نمونهٔ دیگری از گردش‌نما را به دست داده‌ایم. این برنامه مجموع عددهای صحیح را از یک تا عددی که در عبارت داده‌های شماره ۱۰۰ جای گرفته است، به دست می‌آورد. در برنامهٔ زیر عددهای ۱ تا ۵۰ را با یکدیگر جمع کرده‌ایم. شکل ۲ــ ۴ نشان دهنده گردش کار برنامه می‌باشد (برای آسانتر فهمیدن گردش‌نما شمارهٔ سطرها را نیز نوشته‌ایم):

```
    100 DATA 50
    110 READ L
    120 S = 0
    130 C = 1
    140 S = S + C
    150 IF C = L THEN 180
    160 C = C + 1
    170 GO TO 140
    180 PRINT "SUM OF INTEGERS 1 TO "; L; " IS ";S
    190 END
RUN
SUM OF INTEGERS 1 TO 50 IS 1275
```



شکل ۲— ٤ گردش نمای برنامهٔ جمع عددهای صحیح

سطرهای ۱٤۰ تا ۱۷۰ حلقه‌ای می‌سازند که هر بار به اندازهٔ متغیر به مجموع اعداد اضافه می‌نماید. این حلقه تا هنگام که مقدار متغیّر شمارنده (C) برابر متغیّر کنترل (L) شود تکرار خواهد شد. هر حلقه رامی به چند گونه نوشت که انتخاب یکی از آن‌ها بیشتر به سلیقهٔ شخصی بستگی دارد. در زیر یک مسئله را به سه روش گوناگون حل کرده‌ایم.

برنامه زیر که دارای سه نگارش گوناگون می‌باشد اعداد صحیح ۱ تا ٦ را نمایش می‌دهد. در نگارش نخست دستور سنجش پس از نمایش هر عدد انجام می‌شود. نگارش‌های دوم و سوم کار سنجش را پس از افزودن برشمارنده (یعنی متغیر C) انجام می‌دهند. درنگارش دوم دستور سنجش، شرط «بزرگتر از»

را به کار می‌گیرد، و در نگارش سوم شرط «کوچکتر از» را.

نگارش نخست

```
100 C = 1
110 PRINT C
120 IF C = 7 THEN 150
130 C = C + 1
140 GO TO 110
150 END
```

نگارش دوم

```
100 C = 1
110 IF C > 6 THEN 150
120 PRINT C
130 C = C + 1
140 GO TO 110
150 END
```

نگارش سوم

```
100 C = 1
110 PRINT C
120 C = C + 1
130 IF C < 7 THEN 110
140 END
```

توجه نمائید که سه ویرایش دارای سطرهای برابر نیستند. هرچند تعداد سطرهای برنامه مسئله‌ای نیست ولی گاهی می‌توان با به کار بردن حلقه از شمار سطرها کاست. در حقیقت حلقه دارای چنان اهمیّتی است که در زبان بیسیک دستور ویژه‌ای برای ساختن آن وجود دارد که آن را دستور «تکرار» می‌نامیم.

دستور تکرار با واژه FOR آغاز شده و با واژهٔ NEXT پایان می‌یابد. دستور تکرار دارای شمارنده‌ای است که در عبارت FOR نام آن،

مقدار نخستین آن، مقدار پایانی آن و مقدار افزاینده که در هر تکرار باید به آن افزوده شود تعیین می‌شود. عبارت NEXT پایان دستور تکرار را مشخص می‌نماید. هنگامی این عبارت انجام می‌شود کنترل به عبارت FOR مربوط به آن جهش می‌کند مگر اینکه مقدار شمارنده از مقدار پایانی بزرگتر شده باشد. برنامهٔ زیر همان برنامهٔ قبلی است که این بار آن را با کمک دستور تکرار نوشته ایم:

```
100 FOR C = 1 TO 6 STEP 1
110 PRINT C
120 NEXT C
```

سطر شماره ۱۰۰ دستور تکرار می‌باشد. شمارندهٔ این دستور متغیر C است که مقدار نخستین آن ۱ و مقدار پایانی آن ٦ می‌باشد. مقدار افزاینده (STEP) ۱ می‌باشد. با این دستور حلقه ٦ بار تکرار می‌شود و متغیّر C به ترتیب مقدارهای ۱، ۲، ۳، ٤، ۵، ٦ را دارا خواهد بود. در بیشتر گونه های بیسیک در صورتیکه مقدار افزاینده ۱ باشد می‌توان آن را حذف کرد. از این رو می‌توان سطر ۱۰۰ را به صورت زیر نوشت:

```
100 FOR C = 1 TO 6
```

سطر ۱۱۰ مقدار C را در هر تکرار نمایش خواهد داد. در سطر ۱۲۰ که پایان حلقه را مشخص می‌نماید مقدار تازهٔ شمارنده، با افزودن افزاینده به آن، محاسبه می‌شود. اگر شمارنده از مقدار پایانی بزرگتر نباشد حلقه دوباره تکرار می‌شود (در این مثلاً اگر C از ٦ بزرگتر نباشد).

در دستور تکرار می‌توان برای شمارنده مقدارهای مختلفی درنظر گرفت، به شرط اینکه با این مقدارها حلقه درستی ساخته شود:

```
240 FOR A = B TO 10 STEP 6
300 NEXT A
340 FOR G = 24 TO 2 STEP - 2
390 NEXT G
460 FOR B = 1 TO R*3
500 NEXT B
```

توجه نمائید که برای هر عبارت FOR باید یک

عبارت NEXT وجود داشته باشد. اگر جز این باشد برنامه «نا به هنگام» با فرستادن پیغام زیر به پایان میرسد:

FOR WITHOUT NEXT IN...

مقدار متغیر کنترل یا شمارنده را نباید در داخل حلقه تغییر داد. هرچند، چنان که دیدیم، می‌توان آن را در محاسبه به کار برد.

کار درستی نیست که از حلقه تکرار با کمک دستورهایی همچون جهش و اگر– پس به بیرون جهید. همچنین جهیدن به درون حلقه احتمالاً در کار برنامه ایجاد اشکال می‌کند (FOR WITHOUT NEXT و جز آن) باید با عبارت FOR وارد حلقه شده و پس از پایان تکرار، به وسیلهٔ عبارت NEXT از آن خارج شد.

برنامه نمونهٔ زیر میانگین چهار گروه عدد را محاسبه می‌کند. عددها و میانگین‌ها در پایان برنامه نشان داده می‌شوند:

```
100 DATA 20,16,44,18,20
110 DATA 18,12,19,14,23
120 DATA 40,16,12,17,34
130 DATA 23,15,19,29,37
200 T = 0
210 FOR M = 1 TO 5
220 READ N
230 PRINT N;
240 T = T + N
250 NEXT M
260 PRINT "AVERAGE = ",T/5
270 GO TO 200
999 END


RUN

20 16 44 18 20 AVERAGE = 23.6
18 12 19 14 23 AVERAGE = 17.2
40 16 12 17 34 AVERAGE = 23.8
23 15 19 29 37 AVERAGE = 24.6
OUT OF DATA IN LINE 220
```

از آنجا که می‌دانیم چهار دسته «داده» داریم، می‌توانیم با افزودن یک حلقهٔ دیگر به برنامه کاری کنیم که محاسبه‌های اصلی تنها چهار بار انجام شوند. با این کار برنامه به شیوهٔ درست پایان می‌یابد، زیرا از اجرای دستور خواندن، در حالی که داده‌ای وجود ندارد، جلوگیری می‌شود:

```
100 DATA 20,16,44,18,20
110 DATA 18,12,19,14,23
120 DATA 40,16,12,17,34
130 DATA 23,15,19,29,37
150 FOR X = 1 TO 4
200 T = 0
210 FOR N = 1 TO 5
220 READ M
230 PRINT N,
240 T = T + N
250 NEXT M
260 PRINT "AVERAGE = ", T/5
270 NEXT X
999 END
```

در این برنامه نمونه‌ای از حلقه‌های تودرتو آمده است. حلقه‌ها را می‌توان به تعداد زیادی در داخل هم قرار داد، ولی هر حلقه باید کاملاً در داخل حلقهٔ قبلی قرار گیرد، یعنی عبارت FOR آن پس از عبارت FOR حلقهٔ نخستین و عبارت NEXT آن بیش از عبارت NEXT آن حلقه قرار گیرد. به سخن دیگر حلقه‌ها نباید یکدیگر را قطع نمایند. به عنوان نمونه، دستورهای زیر درست می‌باشند:

```
110 FOR J = 1 TO 10
120 FOR K = 2 TO 6
.
.
.
.
.
200 NEXT K
220 NEXT J
```

در حالی که دستورهای زیر نادرست می‌باشند:

```
110 FOR J = 1 TO 10
120 FOR K = 2 TO 6
.
.
.
200 NEXT J
220 NEXT K
```

در برنامهٔ زیر دو حلقه تودرتو به کار رفته است در حالی که حلقه دوّمی از متغیر کنترل یا شمارنده حلقه اوّلی برای تعیین تعداد تکرارها استفاده کرده است:

```
210 FOR J = 1 TO 10
220 FOR K = 1 TO J
230 PRINT K;
240 PRINT " ";
250 NEXT K
260 PRINT
270 NEXT J
280 END

RUN

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
```

توجه نمائید که سطر ۲۶۰ باعث می‌شود که پاسخ‌ها در سطر بعدی چاپ شوند. سطر ۲۴۰ باعث می‌شود که پاسخ‌ها به دنبال یکدیگر و در یک

سطر چاپ شوند، زیرا بعد از نام متغیر نشان نقطه ویرگول گذاشته‌ایم.

برنامهٔ بعدی از حلقه‌های تودرتو استفاده کرده است ولی نه به صورت افزاینده بلکه به شکل کاهنده:

```
330 FOR J = 1 TO 10
340 FOR K = 10 TO J STEP − 1
350 PRINT K;
360 PRINT " ";
370 NEXT K
380 PRINT
390 NEXT J
400 END

RUN

10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2
10 9 8 7 6 5 4 3
10 9 8 7 6 5 4
10 9 8 7 6 5
10 9 8 7 6
10 9 8 7
10 9 8
10 9
10
```

بار دیگر، در بخش هفتم، هنگامی که «رشته‌ها» را توضیح دهیم از حلقه سخن خواهیم گفت.

بازهم درباره دستور سنجش «اگر– پس»

شکل دستور اگر– پس را که در آغاز این بخش شرح داده شد، چنان انتخاب کردیم که تقریباً در همه دستگاهها بتوان آن را به کار برد. بیشتر گونه‌های معمولی بیسیک دارای دستور «**اگر** شرط **پس** شماره سطر» می‌باشند. در برخی از بیسیک‌ها می‌توان همچنین به جای شماره سطر یک عبارت را به کار برد. برای نمونه، فرض کنید برنامه‌ای بخواهد چنانچه مقدار متغیر A از ۱۰ بزرگتر باشد، به متغیر C عدد ۱ را بیافزاید. می‌توان مسئله را چنین حل کرد:

```
120 IF A < 11 THEN 140
130 C = C + 1
140 . . .
```

و نیز می‌توان به جای دستورهای بالا تنها یک دستور به کار برد:

```
120 IF A > 10 THEN C = C + 1
```

حال چه شرط برقرار باشد و چه نباشد (A از ۱۰ بزرگتر باشد یا نباشد) برنامه پس از این دستور از سطر بعدی کار را دنبال خواهد کرد.

ترکیب دستورهای اگر ــ پس و دستور چاپ از شمار سطرهای برنامه بسیار خواهد کاست. این مطلب را در مثال زیر، که بخشی از یک برنامهٔ بازی است، نشان داده‌ایم. متغیّر N تعداد حرکت‌هایی را که بازی‌کن برای به‌پایان رساندن بازی انجام داده است دارا می‌باشد:

```
660 IF N < 10 THEN PRINT "EXCELLENT SCORE"
670 IF N > 50 THEN PRINT "YOU NEED PRACTICE"
```

چنانچه کار بازی‌کن درحد معقولی باشد (یعنی تعداد حرکت‌ها میان ۱۰ تا ۵۰ باشد) در این صورت پیغامی فرستاده نمی‌شود. در این مثال تعداد پیغام‌ها هیچوقت از یکی بیشتر نخواهد بود. یعنی مقدار N هرچقدر باشد یا پیغامی فرستاده نمی‌شود، یا تنها یک پیغام چاپ می‌شود. باید دقت کرد که اشتباه زیر پیش نیاید:

```
660 IF N > 50 THEN PRINT "YOU NEED PRACTICE"
670 IF N > 20 THEN PRINT "NOT BAD"
```

زیرا برای بعضی از مقدارهای N ، مثلاً ۷۵، هردو پیغام برروی صفحه نمایش داده خواهد شد.

در زیر نمونهٔ دیگر از کاربرد دستور اگر ــ پس را نشان داده‌ایم. در این عبارت‌ها، دستمزد کارمندان را با درصدهای فوق‌العادهٔ مختلف محاسبه کرده‌ایم. نام کارمندان در متغیّر N$ قرار دارد:

```
440 IF N$ = "WILLIAMS" THEN T = T*1.1
450 IF N$ = "GOULD" THEN T = T*1.15
460 IF N$ = "ASTON" THEN T = T*1.1
470 IF N$ = "ROSE" THEN T = T*1.05
```

مقدار فوق‌العاده برای «ویلیامز» ۱۰ درصد، برای «گولد» ۱۵ درصد و... می‌باشد. اگر N$ دارای نامی باشد که در برنامه مشخص نشده است، در این صورت محاسبه‌ای انجام نمی‌شود و به‌آن کارمند فوق‌العاده‌ای داده نمی‌شود (یعنی مقدار T تغییر نمی‌کند).

دستور اگر—پس را می‌توان همچنین برای وارسی دامنهٔ یک متغیر به‌کار برد. با این کار می‌توان بررسی کرد که آیا مقدار یک متغیّر میان دو عدد قرار دارد یا نه، چنانچه قرار ندارد آن را برابر حد بالا یا پائین دامنه قرار داد:

```
120 IF R < 0 THEN R = 0
130 IF R > 60 THEN R = 60
```

در این مورد هرچند هردو دستور انجام خواهد شد ولی تنها یکی از دو شرط می‌تواند برقرار باشد، از این‌رو مشکلی پیش نخواهد آمد.

در بعضی از گونه‌های بیسیک می‌توان واژه‌های اضافی و گوناگون را با دستور اگر—پس به‌کار برد. سه عبارت زیر کار یکسانی را انجام می‌دهند:

```
100 IF A = 0 THEN 200
100 IF A = 0 THEN GO TO 200
100 IF A = 0 GO TO 200
```

دستورهای دیگر سنجش

در برخی از گونه‌های بیسیک می‌توان دستور «جهش وابسته» (ON-GO TO) را به‌کار برد که کار یک کلید چند راهه را انجام می‌دهد. با این دستور، کنترل برنامه، بسته به‌مقدار یک متغیر یا عبارت محاسبه‌ای، به‌یکی از سطرهای برنامه منتقل می‌شود. برای نمونه، در سطر شماره ۴۴۰ کاربرد دستور «جهش وابسته» را نشان داده‌ایم:

440 ON S GO TO 100, 110, 120

در این مثال جای جهش، وابسته به‌مقدار S می‌باشد. چنانچه S برابر ۱ باشد به‌سطر ۱۰۰ و چنانچه ۲ باشد به‌سطر ۱۱۰ و... جهش خواهد شد. (در حالت، اوّل دستور بالا برابر دستور GOTO 100 و در حالت دوّم برابر دستور GO TO 110 و... خواهد بود). اگر مقدار S کوچکتر از ۱ و بزرگتر از تعداد شمارهٔ سطرهای دستور (در این مثال، ۳) باشد کار جهش انجام نخواهد شد و برنامه دستور بعدی را انجام خواهد داد.

چنان‌که در سطر شماره ۵۶۰ نشان داده‌ایم، می‌توان در دستور «جهش وابسته» یک عبارت محاسبه‌ای را به‌کار برد:

560 ON (A*2) + 1 GO TO 770, 870, 420

در این دستور، هربار، مقدار عبارت (A*2) + 1 محاسبه شده و سپس بسته به‌این که نتیجه محاسبه ۱، ۲ یا ۳ باشد به‌سطرهای ۷۷۰، ۸۷۰، ۴۲۰ جهش خواهد شد.

هنگام استفاده از عبارت‌های محاسبه‌ای باید ویژگی بیسیکی را که به‌کارمی‌برید، در استفاده از مقدارهای اعشاری، در نظر داشته باشید. برای نمونه، فرض کنید که هنگام اجرای دستور زیرا مقدار S برابر ۳ باشد:

260 ON S*.9 GO TO 290,340,610,500

نتیجه عبارت محاسبه‌ای این دستور برابر ۲/۷ خواهد بود. برخی از گونه‌های بیسیک، تنها قسمت صحیح عدد را برمیدارند (یعنی ۲ را) در نتیجه به‌سطر

۳۴۰ جهش می‌کنند، و بعضی آن را گرد می‌نمایند (یعنی آن را به ۳ تبدیل می‌کنند) و در نتیجه به سطر ۶۱۰ جهش می‌نمایند.

در مثال زیر یکی دیگر از کاربردهای دستور «جهش وابسته» را نشان داده‌ایم. فرض کنید که متغیر I همچون نماینده‌ای برای تعیین عنوان افراد به‌کار رود. در این بخش از برنامه خواسته‌ایم که با استفاده از دستور جهش وابسته، قبل از نام خانوادگی هر فرد واژه‌ای مناسب چاپ کنیم (مثلاً: آقا، خانم، دوشیزه و...) بازهم فرض کنید که در بخش قبلی این برنامه، بررسی شده است که I حتماً دارای مقداری میان ۱ تا ۵ باشد و چنانچه عنوان فرد مشخص نباشد I را برابر صفر قرار دهد تا عنوانی چاپ نشود.

```
420 ON I GO TO 450,470,490,510,530
430 PRINT TAB(6)
440 GO TO 540
450 PRINT " MR    ";
460 GO TO 540
470 PRINT " MRS   ";
480 GO TO 540
490 PRINT " MISS  ";
500 GO TO 540
510 PRINT " DR    ";
520 GO TO 540
530 PRINT " PROF ";
540 PRINT S$
```

اگر بخواهیم بسته به مقدارهای مختلف متغیر یا عبارت محاسبه به یک سطر جهش شود، باید شمارهٔ آن سطر را به تعداد مورد نیاز تکرار نمائیم:

```
600 ON A GO TO 100,120,100,160,120
```

بخش پنجم

نگهداری برنامه

تا این جای کتاب توانایی لازم برای نوشتن برنامه‌های کوچک را به دست آورده‌ایم. بسیار پیش می‌آید که برای نوشتن یک برنامه باید آن را نخست به بخش‌هایی تقسیم کرد سپس هر بخش را جداگانه نوشت و در کامپیوتر مورد آزمایش قرار داد. برای این که از ماشین کردن دوبارهٔ بخش‌های تمام شده برنامه جلوگیری کنیم، و نیز برای این که یک برنامهٔ کامل را بتوانیم هر گاه که بخواهیم به اجراءِ در آوریم، بی آن که دوباره آن را به کامپیوتر بدهیم، به نوعی «انباره*» برای نگهداری آن‌ها نیاز داریم.

برنامه‌ای را که به کامپیوتر داده‌ایم می‌توانیم چندین بار مورد استفاده قرار دهیم، ولی همین که دستگاه را خاموش کردیم برنامه نیز از میان خواهد رفت. از این رو باید انباره‌ای خارج از کامپیوتر، ولی در دسترس آن، برای نگهداری برنامه داشته باشیم. دو نوع معمولی این انباره‌ها نوار ضبط صوت (کاست) و صفحهٔ لرزان (دیسکت) می‌باشد.

ارزانترین روش برای انبار کردن برنامه به کار بردن دستگاه ضبط می‌باشد که می‌تواند با باطری یا برق شهر کار کند. برای این کار می‌توان از دستگاههای معمولی استفاده کرد و لازم نیست که حتماً دستگاه پخش کاملی را به کار برد. نوارهای معمولی برای این کار مناسب هستند، هر چند تنها بخش کوچکی از آن‌ها مورد استفاده قرار می‌گیرد. از این رو می‌توان از نوارهای چند دقیقه‌ای نیز برای این کار استفاده نمود.

دانستن این که برنامه و داده‌ها با چه روشی بر روی نوار انبار می‌شوند بسیار سودمند است. برای این کار باید نخست دریابیم که اطلاعات در داخل کامپیوتر چگونه نشان داده می‌شوند.

در درون کامپیوتر هرچیز— داده‌ها، برنامه‌ها، ثابت‌ها، عددها و...— به رشته‌ای از رقم‌های «دودویی*» یا «بیت*» تبدیل می‌شوند. هر بیت یا رقم دودویی تنها می‌تواند دو مقدار را بپذیرد، صفر یا یک را. برای نشان دادن اطلاعات گوناگون باید صفرها و یک‌ها را با یکدیگر ترکیب کرد. از این رو حرف A را می‌توان به شکل زیر نشان داد:

0100 0001

توجه کنید برای یک نشانه (مثلاً A) از ۸ بیت استفاده شده است. این هشت بیت را می‌توان به دو دسته چهار بیتی بخش کرد. با این کار می‌توانیم در هنگام نوشتن روش کوتاهتری را برای نشان دادن رشته‌های بیتی به کار بریم. هر دستهٔ چهار بیتی را می‌توان، با صفر و یک کردن رقم‌های چهارگانهٔ آن، به ۱۶ شکل مختلف تبدیل کرد؛ از ۰۰۰۰ تا ۱۱۱۱. هر یک از این حالت‌های ۱۶ گانه را می‌شود با یک رقم یا یک حرف نشان داد:

نشانهٔ هم ارز	دستهٔ چهاربیتی
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

خوانندگانی که با ریاضیات آشنائی دارند می‌دانند که این همان روشی است که برای نشان دادن اعداد به کار می‌رود. ولی در این روش به جای این که از پایهٔ ۱۰ معمولی در دستگاه دهدهی استفاده کرده باشیم، پایهٔ ۱۶ را به کار برده‌ایم.

عددهایی را که در پایهٔ شانزده نشان می‌دهیم «شانزده شانزدهی.*» می‌نامیم.

گفتیم که هر رشتهٔ ۸ بیتی را می‌توان به دو دسته چهار بیتی بخش کرد، و اگر روش بالا را نیز به کار بریم می‌توانیم آن را با دو رقم شانزده‌شانزدهی نشان دهیم. مثلاً ۰۱۰۰ ۰۰۰۱ (یا حرف A) را می‌توان به صورت ۴۱ نشان داد. برای این که این ۴۱ با ۴۱ دهدهی اشتباه نشود آن را به شکل $(۱۶)^{۴۱}$ می‌نویسیم.

در زیر نمونه‌ها بیشتری از رشته‌های بیتی و معادل شانزده‌شانزدهی آن‌ها را آورده‌ایم:

عدد شانزده‌شانزدهی هم ارز	رشتهٔ بیتی
C2	1100 0010
94	1001 0100
1E	0001 1110
03	0000 0011
69	0110 1001
AD	1010 1101

هر برنامه‌نویس بیسیک، که به کار خویش دلبسته باشد، آشنایی با اعداد شانزده‌شانزدهی را سودمند خواهد یافت. فرض کنیم که برای کنترل یک شبکه راه‌آهن بخواهیم از یک ریز کامپیوتر استفاده کنیم. جریان اطلاعات دوسویه است؛ ریز کامپیوتر باید وضعیّت کلیدها و دستگاههای حس کنندهٔ گوناگون را دریافت داشته و بر اساس آن در دوراهی ها مسیر خاصّی را تعیین کرده و علامت لازم را بفرستد. اطلاعات ورودی و خروجی از «درگاههای*» هشت بیتی دریافت و فرستاده می‌شوند. فرض کنیم که ۸

بیت درگاه خروجی نشان دهندهٔ دوراهی ها و چراغ‌ههای علامت دهندهٔ زیر باشند (بیت صفر نخستین بیت دست راست و بیت ۷ نخستین بیت دست چپ می‌باشد):

بیت ۷	محل تقاطع خط ها
بیت ٦	دوراهی ٤
بیت ۵	دوراهی ۳
بیت ٤	چراغ ج
بیت ۳	چراغ ب
بیت ۲	چراغ الف
بیت ۱	دوراهی ۲
بیت ۰	دوراهی ۱

(در همهٔ موارد ۱ به معنی روشن یا باز و ۰ به معنی خاموش یا بسته می‌باشد.)

فرض کنید که در مرحلهٔ خاصّی برنامه بخواهد دوراهی های ۲ و ۳ را باز و چراغ های الف و ج را روشن کند. برای این منظور باید بیت‌ها را به شکل زیر روشن و خاموش کرد:

0
1
1
0
1
1
0
0

ترکیب بالا را می‌توانیم به سادگی با عدد شانزده‌شانزدهی ٦C نشان دهیم. به همین گونه، اگر بخواهیم مثال بالا را چنان تغییر دهیم که دوراهی

۲ بسته شود، کافی است که عدد شانزده‌شانزدهی C۲ را به درگاه خروجی روانه کنیم.

فرستادن مقدارها به یک درگاه خروجی و دریافت وضعیّت از درگاه ورودی معمولاً به وسیلهٔ زبان‌های گسترده‌تر از زبان بیسیک (یعنی زبان اسمبلر) انجام می‌شود. به کار بردن این زبان را در بخش دوازدهم شرح خواهیم داد، هرچند دستگاههای گوناگون دارای درگاههای ورودی و خروجی یکسانی نیستند. خواننده باید در این مورد به کتاب راهنمای ویژهٔ دستگاه خویش مراجعه نماید.

انبارهٔ بیرونی— نوار ضبط یا کاست

طرز نگهداری داده‌ها و برنامه‌ها را، بر روی نوار ضبط، اکنون که دستگاه شانزده شانزدهی را شرح داده‌ایم بهتر می‌توان فهمید. هر اطلاعی را که بخواهیم بر روی نوار بنویسیم باید آن را به شکل دودویی در بیاوریم. از این رو اگر بخواهیم حرف A را به روی نوار ببریم— که میدانیم معادل شانزده‌شانزدهی آن ٤١ است— باید ترکیب بیتی زیر را در نوار بنویسیم.

۰۱۰۰ ۰۰۰۱

برای نوشتن بیت‌ها بر روی نوار از دو نوای شنیداری با بسامد (یا فرکانس)های مختلف استفاده می‌شود. مثلاً، در روش کامپیوتر اَپِل، بسامد ۲ کیلو هرتز برای نشان دادن صفر و ۱ کیلو هرتز برای نشان دادن ۱ به کار می‌رود. دامنهٔ هر نوا در روی نوار به مغناطیس شدن ذرّه‌ها بستگی دارد. چنانچه سرعت یک دستگاه فرضی ۳۰۰ ذرّه در ثانیه باشد، دامنهٔ هر نوا برابر ۲دهم ثانیه خواهد بود. خواندن داده‌ها از روی نوار به وسیلهٔ صافی های کوک شده* یا حلقه‌های زمانی نرم افزاری انجام می‌شود.

برای فرستادن برنامه به کاست فرمان نگهداری کردن (SAVE) را ماشین می‌کنیم:

SAVE

این فرمان همهٔ برنامه را بر روی نوار می‌نویسد. برای این کار باید دستگاه ضبط را به شیوه‌ای که در کتاب راهنمای کامپیوتر شرح داده شده است به آن وصل کرد.

برای خواندن برنامه از نوار، کافی است که ضبط را به کامپیوتر وصل کرده، نوار را به اوّل آن برگردانید و سپس فرمان بار کردن (LOAD) را ماشین کنید:

```
LOAD
```

در بیشتر دستگاهها، تنها می‌توان یک برنامه را در روی هر نوار نگهداری کرد. در این صورت لازم نیست که برای برنامه نامی تعیین نمائیم. ولی در دستگاههایی که می‌توان در روی یک نوار چندین برنامه را نگهداری کرد باید هنگام نگهداری و «بازیابی*» برنامه نام آن را نیز به کار برد:

```
SAVE "PROGA"
LOAD "PROGB"
```

چنان که می‌بینید این فرمان‌ها دارای شمارهٔ سطر نیستند زیرا معمولاً از آن‌ها در داخل برنامه استفاده نمی‌شود.

انبارهٔ بیرونی— صفحهٔ لرزان یا دیسکت

یکی از نقص‌های نوار کاست کمی سرعت آن است. داده‌هایی که بر روی نوار کاست انبار شده باشند تنها به صورت پیوسته دسترس‌پذیر هستند. یعنی برای بازیابی رشته‌ای از داده‌ها باید داده‌های پیش از آن همگی خوانده شوند. هر دوی این نقص‌ها را می‌توان با نگهداری داده‌ها بر روی دیسکت از میان برداشت، هرچند برای این کار باید بهای بشتری پرداخته شود. دیسکت یا صفحهٔ لرزان— این نام را به علت نرمی صفحه به آن داده‌اند— صفحه‌ای است دایره‌ای شکل به قطر ۵/۲۵ اینچ (یا ۸ اینچ در دستگاههای بزرگ بازرگانی) که از ماده‌ای پوشانده شده است که می‌تواند آهنربا شود. این

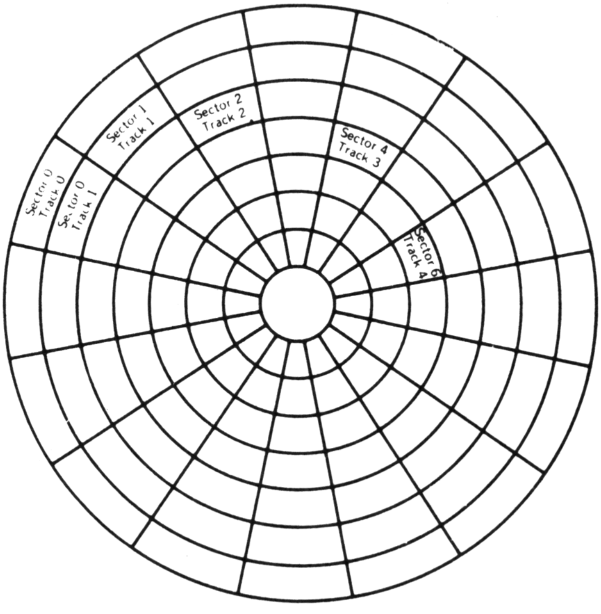
صفحه در داخل پوششی پاکتی شکل قرار گرفته است که در آن شکافی هست که نوک شاخک دستگاه دیسک‌خوان، برای خواندن و نوشتن داده‌ها، در آن جای می‌گیرد.

دیسکت‌ها دارای استانداردهای گوناگونی هستند. در بعضی از هر دو طرف صفحه استفاده می‌شود و در برخی از یک طرف. برخی دارای «فشردگی دوگانه» هستند و بعضی نیستند. ولی از نگاه استفاده کننده این تفاوت‌ها چندان اهمیتی ندارند و می‌توان از آن‌ها و نیز از سازمان دیسک و طرز قرار گرفتن اطلاعات بر روی آن چشم پوشید. این ویژگی‌ها رامی‌توان به «سیستم عامل دیسکی *» (DOS) واگذار کرد که با هر دستگاهی که از دیسکت استفاده می‌کند همراه می‌باشد. (توجه داشته باشید که سیستم عامل دیسکی کارخانه‌های گوناگون یکنواخت و همساز نیستند).

در هر دیسکت، داده‌ها بر روی دایره‌هایی هم‌مرکز که آن‌ها را «کِشه *» (track) می‌نامیم نوشته می‌شوند. هر کِشه به چند بخش تقسیم می‌شود که این بخش‌ها را «بُرِش *» (sector) نام می‌نهیم. در هر خواندن یا نوشتن یکی از این برش‌ها خوانده یا نوشته می‌شود. هر برنامه ممکن است چند برش را که در یک یا چند کشه جای گرفته‌اند به کار گیرد. (شکل ۱—۵ سازمان‌بندی دیسکت را نشان می‌دهد.)

کشه‌ها و برش‌ها از شماره صفر به بالا شماره‌گذاری شده‌اند. نخستین برش هر کشه (یعنی برش شماره صفر) را «برشِ خانه» نام نهاده‌اند. دستگاه دیسکت‌خوان می‌تواند با استفاده از نشانه‌ای ویژه یا سوراخی که بر روی دیسکت قرار دارد برش خانهٔ هر کشه را پیدا کند. سیستم عامل (DOS) معمولاً در برش صفر کشه صفر هر دیسکت «فهرست راهنما» را می‌نویسد که مشخصات همهٔ «پرونده *»های آن دیسکت را در خود دارد. هنگامی که فرمان بازیابی یا بار کردن یک برنامه (یا پرونده) داده می‌شود، فهرست راهنما خوانده شده و نام پرونده در آن پیدا می‌شود. همراه با

نام پرونده اندازه و جای آن بر روی دیسکت نیز خوانده می‌شود. سپس سیستم عامل نوک شاخک خواندن و نوشتن را راهنمایی می‌کند تا در کشه و برشی که پرونده از آن جا آغاز می‌شود جای گرفته و آن را بخواند. از این رو لازم نیست که برای یافتن یک پرونده همهٔ دیسکت خوانده شود. این ویژگی، همراه با سرعت بیشتر، دیسکت را نسبت به کاست کارایی بیشتری بخشیده است.



شکل ۱ــ ۵ نمونهٔ سازمان‌بندی دیسکت که در آن شمارهٔ برش‌ها و کشه‌ها نشان داده شده است.

هر سیستم عامل دیسکی دارای فرمان هایی می‌باشد که توانایی کار با دیسکت را به برنامه‌نویس می‌دهد. در زیر نمونه‌هایی از فرمان‌های سیستم عامل آپل (Apple) را می‌آوریم (این فرمان‌ها را «فرمان‌های خانه‌داری*» می‌نامیم).

CATALOG

با این فرمان فهرست راهنمای دیسکت نشان داده می‌شود. چنانچه شماره دستگاه مشخص نشود، دستگاه شمارهٔ صفر برگزیده می‌شود. اگر دستگاه دیسک گردان دیگری نیز داشته باشیم و بخواهیم فهرست راهنمای آن را نشان دهیم، باید شمارهٔ آن را نیز یادآوری نمائیم.

پیش از آن که یک دیسکت نو را مورد استفاده قرار دهیم، باید بسته به کامپیوتر، به آن «آمادگی آغازین*» داده شود. این پردازش از آماده کردن فهرست راهنما و شکل‌بندی برش‌ها و کشه‌ها تشکیل شده است. نمونهٔ دستور آماده ساختن آغازین (INIT) چنین است:

INIT START

واژه‌ای که بعد از فرمان آمده است، نام برنامه‌ای است که همراه با آماده ساختن آغازین دیسکت در روی آن انبار می‌شود. هرگاه که از این دیسکت استفاده شود، پیش از هر کار، برنامهٔ نامبرده اجراءِ خواهد شد. نمونهٔ برنامهٔ START در زیر داده می‌شود:

```
10 PRINT "SLAVE DISK SET UP 1.12.82"
20 END
```

چنان که پیش از این گفتیم برای نگهداری و بازیابی (یا انبار کردن و بار کردن) پرونده‌ها از فرمان‌های SAVE و LOAD استفاده می‌شود. برای پاک کردن پرونده‌ای از روی دیسکت فرمان «زدودن» (DELETE) را به کار می‌بریم:

DELETE filename

(بعد از فرمان نام پرونده آمده است). برای این که از پاک شدن تصادفی پرونده‌ای جلوگیری شود باید آن را با فرمان (LOCK) قفل می‌کنیم:

LOCK FILEA

برای پاک کردن چنین پرونده‌ای باید دو فرمان «قفل‌گشایی» (UNLOCK) و زدودن را با هم به کار بریم:

```
UNLOCK FILEA
DELETE FILEA
```

برای تغییر نام یک پرونده در فهرست راهنما باید از فرمان «تغییر نام» (RENAME) استفاده کرد.

```
RENAME FILEA,FILEB
```

با این فرمان پرونده‌ای که به نام FILE A شناخته می‌شد، بعد از این با نام FILE B شناخته می‌شود.

فرمان‌های بالا در دستهٔ پردامنه‌ای از فرمان‌ها، که به نام «فرمان‌های خانه‌داری» خوانده می‌شوند، جای می‌گیرند. این فرمان‌ها به برنامه‌نویس این توانایی را می‌دهند که برنامه‌های دیسکی خود را نگهداری و رهبری نمایند، از این رو در شمار فرمان‌های «همزمان» می‌باشند (زیرا خارج از برنامه به کار می‌روند).

پرونده‌هایی که از داخل برنامه مورد استفاده قرار می‌گیرند (با دستورهایی که بعد خواهیم دید) بیشتر برای نگهداری اطلاعات به کار می‌روند تا برنامه. برای نمونه، جدول زیر نشان دهندهٔ انواع برنامه‌ها همراه با پرونده‌های موردنیاز هر کدام می‌باشد:

نوع برنامه	پرونده‌های دیسکی مورد نیاز ورودی	خروجی
پرداخت حقوق	مشخصات کارمندان	مشخصات کارمندان
	دستمزدها	
چاپ صورت حساب	نشانی مشتریان	
	فروش	فروش
		حساب های سررسیده
طراحی مدار	بلوک های ساختمانی	
	اندازه های استاندارد	
حساب های منزل	فعالیت های جاری	
	پروندهٔ شخصی	پروندهٔ شخصی
واژه پردازی	متن های استاندارد	
	نوشته ها	نوشته ها
	نشانی مشتریان	
بازی	جدول کارایی بازی کنان	جدول کارایی بازی کنان
	نقشه راه مار پیچی	
	طرح «ستاره»ای	

توجه نمائید که در بعضی موارد یک پرونده هم به عنوان ورودی و هم به عنوان خروجی، به کار رفته است. این از آن رو است که به هنگام اجرای برنامه، اطلاعات برخی از برنامه ها باید تازه شود، یعنی این پرونده‌ها «به هنگام» (update) شوند. موضوع را با مثال زیر بهتر می‌توان شرح داد:

در برنامهٔ پرداخت حقوق، پروندهٔ مشخصات کارمندان خوانده می‌شود. برای هر کارمند اطلاعاتی به شرح زیر وجود دارد:

نام

نشانی

کار

بخش

ساعت یا روز کار

شماره بیمه
درصد فوق العاده
رمز مالیات
درصد پرداخت هفتگی
پرداخت امسال
مالیات امسال
اضافه کاری امسال

هرگاه که برنامه اجراءِ شود سه جزءِ (یا سه میدان Field) پایانی تغییر خواهند کرد. مثلاً به مقدار پرداخت امسال، پرداخت این هفته نیز افزوده خواهد شد، و این عمل دربارهٔ مالیات امسال و اضافه کاری امسال نیز انجام خواهد شد.

نگارش های گوناگون یک پرونده را «نسل»های آن می‌گوئیم. هنگام کار با یک پرونده ترتیب پیدایش نسل ها چنین است: پدربزرگ، پدر، پسر. از این رو در هنگام به پایان رسیدن اجرای هفتگی برنامه سه پرونده دردست خواهد بود. پروندهٔ این هفته، پروندهٔ هفتهٔ پیش و پروندهٔ دو هفته پیش. نگهداری سه نسل از یک پرونده برای این است که چنانچه پروندهٔ این هفته به علتی از میان برود، بتوان آن را با استفاده از نسل پیشین دوباره بازسازی کرد. نسل های پرونده‌های مهم باید روی سه دیسکت جداگانه، و هر کدام در جایی جداگانه، نگهداری شوند.

شاید دارندگان کامپیوترهای خانگی لازم نباشد که چنین احتیاط هایی را کاملاً رعایت کنند. به هر حال، کاری خردمندانه است که از پرونده‌ها و برنامه‌های مهم نسخه‌های دیگری نیز نگهداری شود. دیسکتی که برای نگهداری نسخه‌های «پشتیبانی» به کار می‌رود باید در جای مطمئنی که از گرمی و نمناکی بیش از اندازه برکنار باشد گذاشته شود.

دوباره یادآوری می‌کنیم، تا هنگامی که با ماشین کردن برنامه‌ای را به دستگاه می‌دهیم. هر زمان ممکن است آن را به دلیلی از دست بدهیم. اگر

سوختن «فیوز» یا قطع شدن برق باعث خاموش شدن دستگاه شود، برنامه‌ای که در حافظهٔ کامپیوتر قرار گرفته است از میان می‌رود. هنگام ماشین کردن برنامه های بزرگ، بهتر است که در مرحله های گوناگون، برنامه را تا جایی که فراهم آمده است با به کار بردن فرمان «نگهداری» (SAVE) به روی دیسکت منتقل نمائیم تا از گرفتارهای بعدی جلوگیری کرده باشیم.

در بخش ششم، خواندن و نوشتن پرونده‌ها را از داخل برنامه شرح خواهیم داد.

بخش ششم

ورودی و خروجی برنامه

در بخش سوم عبارت «داده‌ها» و دستور «خواندن» را دیدیم که به وسیلهٔ آن‌ها داده‌های لازم به برنامه داده می‌شود. ولی ممکن است بخواهیم که داده‌های برنامه را در هنگام اجراءِ به آن بدهیم. برای این کار دستور «درون‌داد*» (INPUT) را به کار می‌بریم:

```
200 INPUT N
```

در هنگام اجرای این دستور، بر روی صفحهٔ نمایش علامت سؤالی پدیدار شده و برنامه برای دریافت مقدار N منتظر می‌ماند. برای این کار باید عدد N را ماشین کرده و با فشار دادن دکمهٔ بازگشت (RETURN) آن را وارد کامپیوتر نمائیم. از این رو اگر در برنامه‌ی، زیر، در جلوی علامت سؤال عدد ۲۷/۳ را ماشین کنیم، بعد از آن مقدار N برابر این عدد خواهد بود:

```
10 PRINT "GIVE ME A NUMBER"
20 INPUT N
30 PRINT "YOU GAVE ME ";N
40 END
```

```
RUN

GIVE ME A NUMBER
?27.3
YOU GAVE ME 27.3
```

در دستور «درون‌داد» ممکن است بیش از یک متغیر نام برده شوند:

```
220 INPUT A,B,C
```

سه مقداری را که در پاسخ علامت سؤال به کامپیوتر می‌دهیم باید با ویرگول از هم جدا سازیم. اگر فرضاً در پاسخ دستور شماره ۲۲۰، که سه عدد را می‌خواهد دریافت نماید، تنها دو عدد را ماشین کنیم، کامپیوتر برای دریافت عدد سوم، دوباره علامت سؤال را نمایان خواهد ساخت:

```
?16,14
?13
```

اگر در مثال بالا به‌جای سه عدد چهار عدد یا بیشتر از آن را به کامپیوتر بدهیم، تنها سه عدد اوّل برداشته خواهد شد و اعداد اضافی فراموش خواهند شد. در مثال زیر، که در پاسخ همان دستور ۲۲۰ ماشین شده است، از مقدارهای ۸، ۱۲، ۱۶ چشم‌پوشی خواهد شد:

```
?5,20,2,8,12,16
```

از دستور درون‌داد می‌توان برای دادن مقدارهای حرفی به کامپیوتر نیز استفاده کرد:

```
260 INPUT A$
```

در این مورد باید متنی را که می‌خواهیم به‌دستگاه بدهیم در داخل علامت نقل‌قول گذاشته و بعد از علامت سؤال ماشین نمائیم:

```
?"LONDON"
```

در دستور درون‌داد می‌توان چند متغیر حرفی را با هم به کار برد:

```
270 INPUT C$,G$,H$,Y$
```

و نیز می‌توان متغیرهای حرفی و عددی را با هم به کار برد:

```
290 INPUT A,B$
```

در این مورد باید دقت کرد که مقدارهای داده شده با نوع متغیرها درست همخوانی داشته باشند. مثلاً در پاسخ به‌دستور ۲۹۰، دو مقدار داده شده باید به‌ترتیب عددی و حرفی باشند.

?16,"NAME"

چنانچه میان داده‌ها و متغیرها ناهمخوانی وجود داشته باشد، «وضعیت اشتباه» پیش می‌آید، که بسته به دستگاه ممکن است باعث قطع برنامه شود.

در نمونه های زیر نشان داده‌ایم که چگونه می‌توان دستور درون‌داد را برای کارهای «بِده و بِستان» به کار برد. نخست برنامه‌ای را که در بخش سوّم برای تبدیل فوت به‌متر نوشتیم به یاد بیاورید. اکنون همان برنامه را با استفاده از دستور «درون داد» به‌جای «خواندن» می‌نویسیم (توجه نمائید که در این‌جا برای بهتر شدن پاسخ‌ها، نخست مقدمه‌ای را چاپ می‌کنیم):

```
100 PRINT "CONVERT FEET TO METRES"
110 PRINT
120 PRINT "HOW MANY FEET"
130 INPUT F
140 IF F = 0 THEN 190
150 Y = F/3
160 M = Y*0.9144
170 PRINT F; " FEET EQUALS ";M;" METRES"
180 GO TO 110
190 END
RUN
CONVERT FEET TO METRES
HOW MANY FEET
?12
12 FEET EQUALS 3.66 METRES

HOW MANY FEET
?6
6 FEET EQUALS 1.83 METRES

HOW MANY FEET
?0
```

نمونهٔ دوم برنامه کوچکی است که به‌بررسی «احتمالات» می‌پردازد. این برنامه احتمال ریاضی این پیش آمد را محاسبه می‌نماید که در میان یک گروه، دو فرد دارای تاریخ تولّدِ یکسان باشند. درصد احتمال

به‌تعداد افراد گروه بستگی دارد، که این تعداد از راه دستور درون‌داد به کامپیوتر داده می‌شود:

```
90 P = 1
100 PRINT
110 PRINT "HOW MANY PEOPLE IN GROUP"
120 INPUT N
130 IF N = 0 THEN 999
140 IF N < 0 THEN 110
150 IF N > 365 THEN 110
160 FOR J = 365 TO (365-N) STEP -1
170 P = (P*J)/365
180 NEXT J
190 Q = 1 - P
200 PRINT "THE PROBABILITY OF 2 OR MORE"
210 PRINT "IDENTICAL BIRTHDATES AMONG"
220 PRINT N; " PEOPLE IS "; Q
230 GO TO 90
999 END
RUN
HOW MANY PEOPLE IN GROUP
?30
THE PROBABILITY OF 2 OR MORE
IDENTICAL BIRTHDATES AMONG
30 PEOPLE IS 0.730455

HOW MANY PEOPLE IN GROUP
?0
```

به‌عنوان سومین نمونه بخشی از یک برنامه را آورده‌ایم که نشان می‌دهد چگونه می‌توان یک برنامه را «دوستانه‌تر» نوشت. این چنین برنامه‌هایی، به‌خصوص برای کامپیوترهایی که مورد استفادهٔ تازه‌کاران قرار می‌گیرند، بسیار مهم می‌باشند. کامپیوتر این توانایی را دارد که پیغام‌های خوانا و با معنی بفرستد؛ به‌هنگام دادن و گرفتن اطلاعات باید از به‌کار بردن واژه‌های فنی خودداری کرد. (توجه نمائید که برای بعضی از **پرسش‌ها** پاسخ‌های خاصی در نظر گرفته نشده است.)

```
180 R = 0
190 P = 0
200 PRINT "HELLO, I AM A ZX81 COMPUTER"
210 PRINT "WHAT IS YOUR NAME"
220 INPUT N$
230 PRINT "HAVE YOU PLAYED BEFORE"
240 INPUT Q$
250 IF Q$ = "YES" THEN 310
260 P = 1
270 PRINT "OK "; N$; " IT IS QUITE EASY"
280 PRINT "WOULD YOU LIKE INSTRUCTIONS"
290 INPUT Q$
300 IF Q$ = "YES" THEN 600
310 PRINT "DO YOU WISH TO START ";N$;" OR
      SHALL I"
320 INPUT Q$
330 IF Q$ = "YOU" THEN 350
340 R = 1
350 .
```

چهارمین برنامه را برای دوستداران اخترشناسی نوشته‌ایم. این برنامه بر پایهٔ پارامترهایی که به‌کامپیوتر داده می‌شود، درشت‌نمایی دوربین را محاسبه می‌کند:

```
 90 PRINT
100 PRINT "OBJECTIVE LENS DIAMETER IN MM"
110 INPUT D
120 PRINT "FOCAL RATIO OF OBJECT"
130 INPUT R
140 PRINT "EYEPIECE FOCAL LENGTH IN MM"
150 INPUT E
160 L = D*R
170 X = L/E
190 Q = L - X*E
200 Q = (10*Q)/E
210 IF X >D *2 THEN 290
220 PRINT
230 PRINT "MAGNIFICATION = ";X;".";Q
240 PRINT
```

```
250 PRINT "MORE"
260 INPUT A$
270 IF A$ = "YES" THEN 100
280 GO TO 999
290 PRINT "FOCAL LENGTH TOO SMALL"
300 GO TO 250
999 END
```

فهرست گزینه‌ها

یکی از روش‌های سودمند برای راهنمایی کردن استفاده کنندگان برنامه، به کار بردن «فهرست گزینه‌ها*» (MENUS) می‌باشد. برای این کار بر روی صفحهٔ نمایش فهرستی از گزینش‌های ممکن نمایان می‌شود که در کنار آن شماره‌ای وجود دارد. استفاده کننده برای انتخاب هر گزینه شمارهٔ آن را به کامپیوتر می‌دهد. در یک برنامهٔ پیچیده‌تر، هر گزینه ممکن است خود به فهرست دیگری راهنمایی کند، که باید استفاده کننده گزینهٔ خود را از آن برگزیند.

در «نمایش» زیر روش «دوستانهٔ» کار با فهرست گزینه‌های را نشان داده‌ایم:

```
CORNER SHOP NAME & ADDRESS SYSTEM
            CHOICES
1. LIST ALL NAMES AND ADDRESSES
2. ENTER NEW NAMES AND ADDRESSES
3. DELETE A NAME
4. PRINT NEWSPAPER ORDER
5. AMEND NEWSPAPER ORDER
6. AMEND NAME OR ADDRESS
7. RETURN TO OPERATING SYSTEM
WHICH ONE?
```

در برنامهٔ بعدی نشان داده‌ایم که چگونه می‌توان «فهرست گزینه‌ها» را نمایش داد و پاسخ را گرفته و درستی آن را آزمود (فرض کرده‌ایم که

دستگاه دستور «جهش وابسته» (ON – GO TO) را نداشته باشد).
فرض کنید که صفحهٔ نمایش را برای این مثال پاک کرده باشیم:

```
200 PRINT"      LOCAL ESTATE AGENTS SYSTEM"
210 PRINT
220 PRINT"              CHOICE:"
230 PRINT
240 PRINT"      1. SEARCH FOR SUITABLE PROPERTY"
250 PRINT"      2. DELETE PROPERTY SOLD"
260 PRINT"      3. ADD NEW PROPERTY"
270 PRINT"      4. LIST MORTGAGE RATES"
280 PRINT"      5. ENTER SAVINGS PAYMENT"
290 PRINT"      6. RETURN TO MAIN SYSTEM"
300 PRINT
310 PRINT"      WHICH CHOICE";
320 INPUT A
330 IF A < 1 THEN 200
340 IF A > 6 THEN 200
350 IF A = 5 THEN 640
360 IF A = 4 THEN 720
370 IF A = 3 THEN 840
380 IF A = 2 THEN 890
400 PRINT"      PROPERTY SEARCH"
410 PRINT
420 PRINT"              CHOICE:"
430 PRINT
440 PRINT"      1. HOUSE TO BUY"
450 PRINT"      2. HOUSE TO RENT"
460 PRINT"      3. BUNGALOW TO BUY"
470 PRINT"      4. BUNGALOW TO RENT"
480 PRINT"      5. FLAT"
490 PRINT"      6. BED-SITTER"
500 PRINT"      7. SHOP"
510 PRINT"      8. OTHER"
520 PRINT
530 PRINT "     WHICH CHOICE";
540 INPUT A
550 IF A < 1 THEN 530
560 IF A > 8 THEN 530
570 IF A = 8 THEN 900
```

توجه کنید که اگر از فهرست گزینه‌ها، شماره ۱ (یعنی جستجو برای خرید دارایی مناسب) برگزیده شود، برنامه به‌سطر ٤۰۰ جهش می‌نماید و فهرست گزینه‌های فرعی را برای تعیین نوع دارایی دلخواه نمایان می‌سازد.

فهرست گزینه‌ها را می‌توان در پایان بازی، برای این که بازیگر کار دلخواه خود را برگزیند به کار برد:

```
THANK YOU FOR PLAYING THAT GAME
   WHAT DO YOU WANT TO DO NOW?
1.  PLAY AGAIN
2. REVERSE ROLES AND PLAY AGAIN
3. REPLAY THE LAST GAME
4. SEE MY STRATEGY
5. QUIT
WHICH?
```

برای تعیین دستورهایی که باید بعد از «فهرست گزینه‌ها» انجام شوند، از دستور «درون داد» (INPUT) استفاده می‌کنیم. برای نمونه:

```
660 INPUT R
670 ON R GO TO 220, 190,160,900,999
680 PRINT "THATS NOT A CHOICE, TRY AGAIN";
690 GO TO 660
```

نقطه ویرگولی را که در پایان دستور چاپ، در سطر ٦۸۰، گذاشته‌ایم برای این است که علامت سؤالی که از دستور درون‌داد (سطر ٦٦۰) نمایان می‌شود، در همان سطر پیغام، و به‌دنبال آن، چاپ شود. در زیر نمونهٔ دیگری از این روش را نشان داده‌ایم:

```
    550 PRINT "HOW OLD ARE YOU";
    560 INPUT A
    999 END
RUN
HOW OLD ARE YOU?
```

در هنگام اجرای دستور درون‌داد علامت سؤال نمایان می‌شود، از این رو لازم نیست که در دستور چاپ این علامت را در پایان عبارت پرسشی قرار دهیم.

در بسیاری از گونه‌های زبان بیسیک می‌توان دستورهای چاپ و درون‌داد را با یکدیگر ترکیب کرد.

در این صورت متنی را که میخواهیم چاپ کنیم بعد از دستور درون‌داد و در داخل علامت نقل‌قول قرار می‌دهیم.

بعد از علامت نقطه ویرگول گذاشته و سپس متغیرهایی را که باید پس از دستور درون‌داد نام ببریم با ویرگول از یکدیگر جدا می‌سازیم:

```
640 INPUT "WHICH CHOICE ";A
660 INPUT "WHAT'S YOUR NAME ";N$
880 INPUT "WHAT ARE THE THREE DATES ";P,Q,R
```

روش به کار بردن دستور درون‌داد را در برنامهٔ سادهٔ زیر نشان داده‌ایم. این برنامه پس از آن که مبلغ نخستین، مدّت (بر حسب ماه یا سال) و نرخ بهره را دریافت کرد، بهرهٔ مرکب را محاسبه و چاپ می‌نماید:

```
100 PRINT "COMPOUND INTEREST CALCULATION"
110 PRINT
120 INPUT "WHAT IS STARTING VALUE IN £ ";S
130 IF S = 0 THEN 999
140 INPUT "HOW MANY PERIODS ";N
150 INPUT "WHAT IS INTEREST PERCENTAGE PER
      PERIOD ";I
160 F = S*(1+I)↑N
170 PRINT
180 PRINT "FINAL VALUE AFTER ";N;" PERIODS"
190 PRINT "AT ";I;" PER CENT. IS £";F
200 GO TO 110
999 END
```

پرونده‌های دیسکی

در بخش پنجم با روش نوشتن داده‌ها بر روی دیسکت آشنا شدیم. می‌توان در یک زمان پرونده‌ای را خواند یا نوشت، یا این که همزمان آن را خواند و سپس نوشت، یعنی به هنگام درآورد. چه بخواهیم یک پرونده را تنها بخوانیم یا بنویسیم، یا این که هم بخوانیم و هم بنویسیم روش کار یکی است، هر چند دستگاههای گوناگون ممکن است که شیوه‌های گوناگونی را برای اجرای این روش‌ها به کار گیرند.

پرونده‌های دیسکی وسیلهٔ برنامهٔ «سیستم عامل»، که همراه کامپیوتر فروخته می‌شود، اداره می‌شوند. چنان که پیش از این گفتیم، برنامه‌نویس بر هر پرونده نامی می‌گذارد که سیستم عامل آن پرونده را به این نام می‌شناسد. از این رو این نام باید در هر دیسکت تنها یک بار به کار رفته باشد. مشخصات هر پرونده مانند نام و جا و اندازهٔ آن در فهرست راهنمای دیسکت نوشته می‌شود.

پیش از این که پرونده‌ای را به کار گیریم باید آن را با دستور زیر باز کنیم:

```
200 OPEN FILEA
```

این دستور بررسی می‌نماید که پرونده در دیسکت (یا اگر چند دستگاه دیسکت گردان داشته باشیم، در دیسکت مشخص شده) وجود دارد یا ندارد. پس از این که پرونده باز شد، می‌توان آن را با دستورهای «خواندن» و «نوشتن» خواند و نوشت:

```
400 OPEN FILE2
410 WRITE FILE2
.
.
.
.
600 CLOSE FILE2
610 OPEN FILE2
620 READ FILE2
```

با روشی بالا در یک زمان می‌توان پرونده‌ای را خواند یا نوشت. چنانچه بخواهیم پرونده‌ای را که نوشته‌ایم بخوانیم، باید نخست آن را بسته و سپس باز کنیم. در مثال بالا، نخست پرونده‌ای را به‌نام FILE2 ساخته‌ایم، سپس آن را بسته و دوباره به‌عنوان یک پروندهٔ ورودی باز کرده‌ایم.

دستورهای خواندن و نوشتن تنها کاری که می‌کنند پرونده را برای خواندن و نوشتن آماده می‌سازند. برای انتقال اطلاعات میان دیسکت و کامپیوتر باید، چنان که بیش از این در مورد صفحه کلید و صفحهٔ تلویزیون دیدیم، از دستورهای «چاپ» و «درون‌داد» استفاده کرد. برای نمونه، هنگامی که یک پروندهٔ دیسکی می‌سازیم، اگر برای نوشتن از دستور زیر استفاده کرده باشیم:

```
100 PRINT N$,A,B,C
```

سپس می‌توان آن را دستور زیر خواند:

```
600 INPUT R$,P,Q,R
```

توجه نمائید که نوع داده‌ها، در دو دستور، باید با یکدیگر همخوانی داشته باشند.

دستورهای زیر نیز برای خواندن و نوشتن دیسکت به کار می‌روند:

```
    DIN   DOUT
    INP   OUT
    GET   PUT
INPUT n   OUTPUT N
   READ   WRITE
```

(N شماره دیسکت را نشان می‌دهد)

توضیح گستردهٔ پرونده‌های دیسکی از محدودهٔ این کتاب فراتر می‌رود. هر چند در برنامه‌های ساده، می‌توان روشی را که در بالا شرح دادیم برای «پرونده‌های پیوسته» (Sequential) به کار برد. دستورهای مناسب هر دستگاه را می‌توان از کتاب راهنمای آن به‌دست آورد.

بخش هفتم

رشته ها و آرایه ها

گاهی لازم است که در یک برنامه بر روی رشته‌های حرفی کارهایی انجام داد. بعضی از این کارها عبارتند از تعیین طول رشته، برداشتن بخشی از رشته و تغییر نشانه های یک رشته از یک شکل به‌شکل دیگر. با استفاده از تابع طول (LEN) می‌توان طول یک رشته را به‌دست آورد. (تابع ها را در بخش ۹ بیشتر شرح خواهیم داد):

```
100 A = LEN (N$)
```

پس از اجرای دستور بالا، متغیر A برابر تعداد نشانه های متغیر حرفی N$ خواهد شد (که شامل فاصله ها نیز خواهد بود). حال اگر متغیّر N$ شامل رشتهٔ (WHAT IS YOUR NAME) باشد. پس از اجراءِ دستور شماره ۱۰۰، مقدار A برابر ۱۷ خواهد بود.

فرض کنید که بخواهیم در برنامه‌ای نام‌هایی را که به‌وسیلهٔ دستور درون‌داد دریافت کرده‌ایم، در روی صفحه زیر هم چنان نمایش دهیم که همه از طرف راست با یکدیگر تراز شده باشند. (یعنی آخرین حرف همه آن‌ها زیر هم و در یک ستون قرار گیرند). برای این کار می‌توان از تابع طول (LEN) استفاده کرد (فرض می‌کنیم که حداکثر طول نام‌ها ۲۰ حرف باشد و بخواهیم آن‌ها را از دست راست، در ستون ۲۵، تراز کنیم):

```
210 L = LEN (N$)
220 PRINT TAB (25 − L + 1);N$
```

در بخش ششم دیدیم که چگونه می‌توان نشانه‌ها را با رشته‌ای از بیت‌ها نشان داد. ترکیب این بیت‌ها را نیز می‌توان به‌وسیلهٔ عددهای شانزده شانزدهی یا دهدهی نشان داد. برای مثال، حرف A به‌وسیلهٔ رشتهٔ بیتی که معادل مقدار شانزده شانزدهی ٤١ (یا مقدار دهدهی ٦٥) می‌باشد نشان داده می‌شود. تابع رشته (STR$) یکی مقدار عددی را به‌معادل حرفی آن برمی‌گرداند. برای نمونه، اگر N شامل عدد ١٢٣٤ باشد، دستور زیر رشتهٔ حرفی "١٢٣٤" را در R$ قرار می‌دهد.

```
100 R$ = STR$(N)
```

برعکس تابع رشته، تابع ارزش (VAL) معادل عددی یک رشته حرفی را به‌دست می‌دهد. (یادآوری: چنانچه در داخل رشته نشانه‌هائی‌جز رقم‌های صفر تا ۹ قرار گرفته باشند، نتیجهٔ تابع نامعلوم است.) برای مثال، برای برگرداندن مقدار عددی متغیر R$ در بالا، و قرار دادن آن در متغیّر F چنین می‌نویسیم:

```
120 F = VAL (R$)
```

پیش از آن که برنامه سادهٔ زیر را برای نمونه نشان دهیم، بهتر است که یکی دیگر از تابع‌های کار با رشته‌ها، یعنی تابع میان (MID$) را، بررسی کنیم. این تابع برای بیرون کشیدن یک یا چند نشانه از میان رشته به‌کار می‌رود. برای این کار باید جای نخستین نشانه و تعداد آن‌ها را مشخص نمائیم. از این‌رو اگر بخواهیم از رشتهٔ A$ تنها نشانهٔ سوم آن را برداریم، چنین می‌نویسیم:

```
100 P$ = MID$(A$,3,1)
```

اگر A$ شامل رشتهٔ ABCDEF باشد، پس از اجرای عبارت بالا مقدار P$ برابر C خواهد بود. در مثال زیر، با استفاده از تابع بالا، سه نشانهٔ نخست (دست چپ) یک رشته را حذف کرده‌ایم:

P$ = MID$ (A$,3,LEN(A$) − 3)

از آن‌جا که طول A$ را نمی‌دانیم، از تابع طول برای تعیین تعداد حرف‌هایی که باید برداشته شوند استفاده کرده‌ایم.

در زیر برای نمونه برنامهٔ ساده‌ای را آورده‌ایم که نشان دهندهٔ کاربرد تابع‌هایی که تاکنون شرح داده‌ایم می‌باشد. با به‌کار بردن این تابع‌ها طرز دستکاری کردن رشته‌ها (با وارونه کردن یک رشته) و همچنین انجام عملیات ریاضی روی آن‌ها را (با تبدیل رشته‌ها به‌معادل‌های عددی خود) نشان داده‌ایم:

```
100 P$ = ""
110 INPUT "TYPE A NUMBER 1–9,999";I$
120 I = VAL (I$)
130 IF I < 0 THEN 110
140 IF I > 9999 THEN 110
150 I$ = STR$(I)
160 FOR J = LEN (I$) TO 1 STEP − 1
170 P$ = P$ + MID$ (I$,J,1)
180 NEXT J
190 PRINT
200 PRINT "YOU TYPED ";I$
210 PRINT "REVERSED IS ";P$
220 K = VAL (P$)
230 PRINT
240 PRINT "SUM OF ORIGINAL & REVERSED IS "; K + I
250 PRINT
260 INPUT "AGAIN";Y$
270 IF Y$ = "YES" THEN 110
999 END
```

در برخی از دستگاههای ساده‌تر، به‌جای تابع میان (MID$) ، تابعی به‌کار می‌رود که تنها اولین نشانهٔ یک رشته را حذف می‌کند. از این رو اگر متغیر N$ شامل رشتهٔ SAMPLE باشد، با یک بار اجرای عبارت زیر AMPLE به‌دست خواهد آمد:

```
120 N$ = TL$(N$)
```

با استفاده از تابع‌های «آسکی*» (ASC) و نشانه (CHR$) می‌توان یک نشانه را از حالت حرفی به‌عددی و بالعکس تبدیل نمود.

```
100 A = ASC("A")
```

با اجرای این عبارت متغیر A برابر ۶۵ می‌شود (بیاد داشته باشید که A برابر عدد شانزده شانزدهی ۴۱، عدد دهدهی ۶۵ می‌باشد). عبارت بالا را در بعضی از دستگاهها به‌شکل زیر می‌نویسند:

```
100 A = CODE ("A")
```

نتیجه در هر دو صورت یکسان است.

تابع آسکی (ASC) را می‌توان با متغیّرها نیز به‌کار برد. اگر طول متغیّر از ۱ بیشتر باشد این تابع نخستین نشانهٔ متغیّر را مورد عمل قرار می‌دهد:

```
160 C = ASC(P$)
```

اگر مقدار P$ برابر "HELLO" باشد، پس از اجرای عبارت بالا مقدار C برابر معادل عددی حرف H (که برابر ۴۸ شانزده شانزدهی، ۷۲ دهدهی می‌باشد) خواهد بود.

ASC به‌جای ASC II که خود از حروف اوّل عبارت

American Standard Code for Information Interchange

(یعنی رمز استاندارد آمریکایی برای تبادل اطلاعات) ساخته شده است، به‌کار می‌رود. ASC II رمز استاندارد ریز کامپیوترهای همگانی است. در پیوست ۱ مجموعهٔ نشانه‌های اسکی را به‌دست داده‌ایم در این جدول نشانه‌های اسکی و معادل‌های دهدهی و شانزده شانزدهی آن‌ها را نشان داده‌ایم.

توجه نمائید که برای رشته‌های پوچ (null)، که طول آن‌ها صفر است، با تابع آسکی مقدار صفر برگشت داده می‌شود. اگر رشته‌ای با نشانهٔ تهی (یعنی فاصلهٔ خالی) شروع شود، مقدار دهدهی ۳۲ (یا شانزده شانزدهی

۲۰) برگشت داده می‌شود.

یک مقدار عددی را می‌توان با تابع نشانه (CHR$) به‌صورت نشانهٔ آسکی معادل آن برگرداند:

```
330 P = 43
340 A$ = CHR$(P)
```

پس از اجرای دو عبارت بالا، مقدار متغیّر حرفی A$ برابر «+» خواهد شد.

بررسی جدول نشانه‌های اسکی نشان می‌دهد که حرف‌های کوچک و بزرگ تنها در یک بیت با یکدیگر تفاوت دارند. حرف A بزرگ معادل ٤۱ شانزده شانزدهی (۰۱۰۰۰۰۰۱) و حرف a کوچک معادل ٦۱ شانزده شانزدهی (۰۱۱۰۰۰۰۱) می‌باشد. همین گونه، حرف Z بزرگ برابر 5A شانزده شانزدهی (۰۱۰۱۱۰۱۰) و حرف z کوچک معادل 7A (۰۱۱۱۱۰۱۰) می‌باشد.

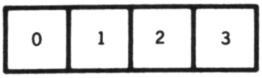
اگر بخواهیم برنامه‌ای بنویسیم که هم حرف‌های کوچک و هم حرف‌های بزرگ را بپذیرد (و چیز دیگری را نپذیرد)، ولی حرف‌های کوچک را هم به‌حرف‌های بزرگ تبدیل نماید به‌روش زیر این کار را می‌توانیم انجام دهیم. این بخش از برنامه هم حرف‌های کوچک را به‌بزرگ برمی‌گرداند و هم وارسی می‌نماید که اگر چیزی جز حرف‌های بزرگ (A−Z) به‌دست آمده باشد، متغیر E را، که به‌عنوان نشانهٔ نادرستی به کار رفته است، برابر ۱ قرار دهد:

```
330 E = 0
340 P = ASC (P$)
350 IF P < 91 THEN 370
360 P = P − 32
370 IF P < 65 THEN 390
380 IF P < 91 THEN 420
390 E = 1
400 P$ = “ ”
410 GO TO. . .
420 P$ = CHR$(P)
```

آرایه‌ها

«آرایه*» (Array) جدولی است از متغیرها که می‌توان آن‌را نوعی قفسه‌بندی به‌شمار آورد. بر آرایه، همچون متغیّرهای معمولی، نامی گذاشته می‌شود. برای دستیابی بر هر «خانه*» آرایه باید نام آرایه را همراه «زیرنویس*» مربوط به آن خانه به کار برد.

شکل ۱—۷ نشان دهندهٔ آرایهٔ ساده‌ای است که از چهار خانه (به شماره‌های ۰، ۱، ۲، ۳) ساخته شده است. اگر نام گذاری آرایه بر عددی بودن آن دلالت کند، آنگاه همهٔ خانه‌های آرایه،



شکل ۱—۷، یک آرایهٔ ساده

همچون دیگر متغیّرهای عددی، می‌توانند تنها مقدارهای عددی را بپذیرند. برای نام گذاری و تعیین تعداد خانه‌های یک آرایه از عبارت «بُعد گذاری*» (DIM) استفاده می‌کنیم:

```
10 DIM A(4)
```

در این عبارت آرایه‌ای را به‌نام A که دارای چهار خانه می‌باشد معرفی کرده‌ایم. این چهارخانه به‌ترتیب دارای نام‌های زیر می‌باشند:

```
A(0)
A(1)
A(2)
A(3)
```

عبارت «بُعد گذاری» را باید قبل از به کار بردن آرایه‌هایی که در آن تعریف شده‌اند معرفی کرد، از این رو بهتر است که این عبارات در آغاز برنامه قرار گیرد. با یک عبارت «بعد گذاری» می‌توان چندین آرایه را معرفی کرد:

```
20 DIM A(4),B(5),C(10)
```

در زیر آرایه‌ای را به‌نام A معرفی کرده‌ایم که دارای پنج خانه می‌باشد، سپس در شکل ۲—۷ نمودار آن را نشان داده‌ایم:

```
10 DIM A (5)
```

هر کدام از این پنج خانه را می‌توانیم به‌وسیلهٔ نام آن‌ها، جداگانه مورد استفاده قرار دهیم.

```
100 A(1) = 2
110 A(2) = B + C
120 R = A(3) + 1
```

شکل ۲—۷

آرایه A و نام خانه‌های آن

A(0)	A(1)	A(2)	A(3)	A(4)

مزیّت واقعی آرایه هنگامی نمایان می‌شود که برای تعیین خانه‌های آن به‌جای شماره داخل پرانتز از متغیّری عددی استفاده کنیم. در مثال سادهٔ زیر نشان داده‌ایم که چگونه می‌توان همهٔ خانه‌های آرایه‌ای را که در عبارت ۱۰ بالا معرفی کرده‌ایم با صفر پر کرد:

```
200 FOR J = 0 TO 4
210 A (J) = 0
220 NEXT J
```

عبارت سطر ۲۰۰ نشان دهندهٔ حلقه‌ای است که مقدار آغازی شمارندهٔ آن (J) برابر صفر و مقدار پایانی آن برابر ٤ در نظر گرفته شده است. چون مقدار افزاینده (STEP) تعیین نشده است، برابر ۱ فرض می‌شود.

در بخش چهارم دیدیم که چنین حلقه‌ای پنج بار تکراری می‌شود و شمارندهٔ آن (یعنی J) به‌ترتیب مقدارهای ۰، ۱، ۲، ۳، ٤ را در برمی‌گیرد. از این رو در مثال بالا سطر ۲۱۰ پنج بار اجراءِ می‌شود، چنان که گویی عبارت‌های زیر اجراءِ شده باشند:

```
A(0) = 0
A(1) = 0
A(2) = 0
A(3) = 0
A(4) = 0
```

در نتیجه آرایه A کاملاً با صفر پر می‌شود.

در عبارت بُعدگذاری (DIM) می‌توان عبارت‌های حرفی را نیز به کار برد:

```
10 DIM A$ (10),R$(6)
.
.
.
100 A$(1) = "HELLO"
110 R$(5) = "      "
```

در نمونهٔ زیر یکی از کاربردهای سادهٔ آرایهٔ حرفی را نشان داده‌ایم. در این برنامه نام ماههای گوناگون را در متغیرهای M$(1) تا M$(12) گذاشته‌ایم (از متغیر M$(0) استفاده نکرده‌ایم تا بتوانیم شماره ماه را مستقیماً از روی تاریخ به‌دست آوریم). پس از روی شماره ماه، نام آن را به‌دست آورده و چاپ می‌کنیم:

```
100 DIM M$(13)
110 DATA "JANUARY","FEBRUARY","MARCH"
120 DATA "APRIL","MAY","JUNE","JULY"
130 DATA "AUGUST","SEPTEMBER","OCTOBER"
140 DATA "NOVEMBER","DECEMBER"
150 FOR J = 1 TO 12
160 READ M$ (J)
170 NEXT J
.
.
200 INPUT "DATE PLEASE DD.MM.YY";D,M,Y
210 IF M < 1 THEN 200
220 IF M > 12 THEN 200
```

```
.
.
300 PRINT "MONTH OF "; M$(M)
.
.
999 END
```

سطرهای ۱۵۰ تا ۱۷۰ آرایهٔ M$ را با نام ماهها پر می‌کنند. در سطر ۲۰۰، برنامه از استفاده کننده می‌خواهد که تاریخی به‌دستگاه داده شود. در سطرهای ۲۱۰ و ۲۲۰ درستی داده‌ها بررسی می‌شود تا شمارهٔ ماه عددی میان ۱ تا ۱۲ باشد. دستور چاپ سطر ۳۰۰، با استفاده از شماره ماه تاریخ، نام ماه را از آرایه برداشته و به‌دنبال واژه‌های داخل علامت نقل‌قول، چاپ می‌کند. فرض کنید که نمونهٔ بالا بخشی از یک برنامهٔ محاسبه فروش ماهانه باشد، و هنگامی که این برنامه را اجراءِ کنیم جمع کل فروش سالانه نیز محاسبه شود. می‌توان برای نشان دادن جمع فروش‌های ماهانه از آرایه T استفاده کرد. پس باید عبارت بُعد گذاری را به‌شکل زیر تغییر داد:

```
100 DIM M$ (13),T(13)
```

در این جا هم برای این که شماره ماه را مستقیماً به‌عنوان زیرنویس به‌کار بریم، برای آرایه ۱۳ خانه در نظر گرفته‌ایم که خانهٔ T(0) به‌کار گرفته نمی‌شود. هنگامی که، در سطر ۲۰۰، برنامه تاریخ را دریافت کند، زیرنویس هر دو آرایهٔ M$ و T تغییر می‌کند. مقدار فروش روزانه را می‌توان بعد از سطر ۲۰۰ خواند و هر کدام را به‌جمع فروش ماهانهٔ مربوط به آن افزود:

```
250 INPUT "DAY'S SALES IN £";S
260 T(M) = T(M) + S
```

از این رو هرگاه که تاریخ به‌دستگاه داده شود مقدار فروش روزانه به‌جمع فروش ماه مربوط به آن روز افزوده می‌شود. با افزایش یک عبارت به‌حلقه‌ای که برای پر کردن آرایهٔ M$ ساخته بودیم، می‌توان آرایهٔ T را نیز همزمان با صفر پر کنیم:

```
150 FOR J = 1 TO 12
160 READ M$(J)
170 T(J) = 0
180 NEXT J
```

بعد از این که تمام مقدارهای فروش روزانه را به‌دستگاه دادیم، با عبارت‌های زیر می‌توانیم جمع کل فروش را محاسبه کرده و چاپ نماییم:

```
500 PRINT "MONTHLY SUMMARY OF SALES"
510 PRINT
520 PRINT
530 G = 0
540 FOR J = 1 TO 12
550 PRINT M$(J);TAB(20);"£";T(J)
560 G = G + T(J)
570 NEXT J
580 PRINT
590 PRINT "TOTAL FOR YEAR"; TAB (20);"£";G
```

یک آرایه را می‌توان به‌صورت وارونه (یعنی از پایان به جلو، یا از دسامبر به ژانویه در مثال بالا) پیمود. برای این کار باید مقدارهای دستور تکرار را تغییر دهیم:

```
540 FOR J = 12 TO 1 STEP - 1
```

آرایه می‌تواند به‌جای یک بعد، که تاکنون دیدیم، دارای چند بعد باشد. در بعضی از گونه‌های بیسیک می‌توان برای آرایه‌ها چندین بعد در نظر گرفت، هر چند برای سادگی بیشتر، ما تنها دو بعد برای آرایه‌ها تعریف خواهیم کرد.

شکل ۳ـ۷ نشان دهندهٔ آرایهٔ دو بعدی A است که در عبارت بعد گذاری زیر تعریف شده است:

```
20 DIM A(4,3)
```

A(0,0)	A(0,1)	A(0,2)
A(1,0)	A(1,1)	A(1,2)
A(2,0)	A(2,1)	A(2,2)
A(3,0)	A(3,1)	A(3,2)

شکل ۳ـ۷ آرایهٔ دوبعدی

چنان که دیدیم عبارت بالا برای معرفی آرایه‌ای که دارای چهار سطر و سه ستون باشد به کار می‌رود (البته در بعضی از بیسیک‌ها با عبارت بالا آرایه‌ای که دارای سه سطر و چهار ستون باشد معرفی می‌شود، هر چند از نظر برنامه‌نویسی تفاوتی میان این دو وجود ندارد). اگر بخواهیم نخست ستون‌های یک آرایه را بپیمائیم روش زیر را به کار می‌بریم:

```
110 FOR J = 0 TO 2
120 FOR K = 0 TO 3
130 PRINT A(K,J)
140 NEXT K
150 NEXT J
```

و اگر بخواهیم نخست سطرها پیموده شوند این روش را:

```
210 FOR J = 0 TO 3
220 FOR K = 0 TO 2
230 PRINT A(J,K)
240 NEXT K
250 NEXT J
```

اگر بخواهیم همهٔ خانه‌های یک آرایهٔ را با صفر (یا دربارهٔ آرایه‌های حرفی با تهی) پر کنیم مهم نیست که اوّل کدام بعد پیموده شود:

```
100 DIM A(2,3),B(3,2)
110 FOR J = 0 TO 2
120 FOR K = 0 TO 3
130 A(J,K) = 0
140 NEXT K
150 NEXT J
160 FOR J = 0 TO 3
170 FOR K = 0 TO 2
180 B(K,J) = 0
190 NEXT K
200 NEXT J
```

عملاً با یک حلقه می‌توان هر دو آرایهٔ بالا را با صفر پر کرد. برای این کار باید متغیرهای زیرنویس (K , J) را جابه‌جا کرد:

```
100 DIM A(2,3),B(3,2)
110 FOR J = 0 TO 3
120 FOR K = 0 TO 2
130 A(K,J) = 0
140 B(J,K) = 0
150 NEXT K
160 NEXT J
```

با جابه‌جا کردن زیرنویس‌ها می‌توان سطرها و ستون‌های یک آرایه را با ستون‌ها و سطرهای آرایهٔ دیگر پر کرد:

```
270 FOR J = 0 TO 3
280 FOR K = 0 TO 2
290 A (K,J) = B(J,K)
300 NEXT K
310 NEXT J
```

آرایه دو بعدی را به خوبی می‌توان در کاربردهایی همچون محاسبهٔ جمع فروش ماهانه به کار برد. فرض کنید برنامه‌ای بخواهد جمع فروش ماهانه کالایی را در چهار منطقه، محاسبه نماید. با استفاده از عبارت بُعد گذاری

(DIM) آرایه‌ای دو بعدی معرفی می‌کنیم (در این برنامه از خانه‌هایی که در زیرنویس آن‌ها صفر به کار رفته باشد، به‌دلیلی که قبلاً شرح دادیم، استفاده نمی‌شود):

```
100 DIM T(13,5)
```

اگر متغیّر M را برای نگهداری شمارهٔ ماه و A را برای منطقهٔ فروش (۱ تا ٤) به کار بریم از عبارت زیر برای افزودن فروش‌های روزانه به‌جمع فروش ماهانه مربوط به آن استفاده می‌کنیم:

```
220 T(M,A)=T(M,A) + S
```

پس از آن که تمام فروش‌های روزانه را به‌دستگاه دادیم می‌توانیم جمع کل فروش هر منطقه را از راههای گوناگون محاسبه کنیم. نمونهٔ زیر یکی از این راه‌ها را برای محاسبه جمع کل فروش سالانه نشان می‌دهد:

```
340 G = 0
350 PRINT
360 FOR J = 1 TO 4
370 Y = 0
380 FOR K = 1 TO 12
390 Y = Y + T(K,J)
400 NEXT K
410 G = G + Y
420 PRINT "ANNUAL SALES FOR AREA ";J;" = £";Y
430 NEXT J
440 PRINT
450 PRINT "GRAND TOTAL = £";G
```

برای چاپ جمع فروش ماهانه در هر منطقه، می‌توان روش زیر را به کار برد (به‌یاد داشته باشید که آرایه M$ در بردارندهٔ نام ماهها می‌باشد):

```
670 PRINT
680 FOR J = 1 TO 12
690 A = 0
700 FOR K = 1 TO 4
710 A = A + T(J,K)
720 NEXT K
730 G = G + A
740 PRINT "TOTAL SALES FOR "; M$(J);" = £";A
```

```
750 NEXT J
760 PRINT
770 PRINT "GRAND TOTAL = £";G
```

توجه نمائید که مقدارهای به‌دست آمده از سطرهای ٤٥٠ و ۷۷۰ یکسان خواهد بود زیرا در هر دو مثال بالا همهٔ خانه‌های آرایهٔ T پیموده می‌شوند. تفاوت تنها در ترتیب پیمودن سطرها و ستون‌های این دو آرایه می‌باشد.

در مثال زیر، آرایه حرفی دو بعدی را برای نگهداری نام اعضاءِ هیئت رئیسهٔ باشگاههای گوناگون به کار برده‌ایم. مثلاً ستون نخست آرایه را برای نگهداری نام رئیس‌های باشگاهها، ستون دوّم را برای منشی های باشگاهها و... در نظر گرفته‌ایم. در هر سطر نام اعضاءِ هیئت رئیسهٔ یک باشگاه را نگهداری کرده‌ایم. در زیر سطرهایی را از یک برنامهٔ نامه‌نگاری آورده‌ایم (آرایه N$ برای نگهداری نام‌ها، و متغیّر C برای شمارهٔ باشگاهها به کار رفته‌اند):

```
1100 PRINT "DEAR MR. ";N$(C,1)
1110 PRINT "AS CHAIRMAN YOU WILL APPRECIATE"
.
.
.
1210 PRINT "YOUR SECRETARY, MR. ";N$(C,2)
```

اگر بخواهیم از نام همهٔ رئیس‌های باشگاهها لیستی فراهم آوریم، روش زیر را به کار می‌بریم (R نشان دهندهٔ تعداد باشگاهها است):

```
1600 PRINT "CLUB CHAIRMEN"
1610 PRINT
1620 FOR J = 1 TO R
1630 PRINT "CLUB NO. ";J;" :-MR. ";N$(R,1)
1640 NEXT J
```

آرایش آرایه‌ها

گاهی لازم است که آرایه‌ها را به‌ترتیب «فرازی» (از کوچک به بزرگ) یا «فرودی» (از بزرگ به کوچک) مرتّب و آرایش* (sort) کنیم. ساده‌ترین روش این کار آن است که «آرایش موجی» نام گرفته است. برای این کار مقدار نخستین خانه یک آرایه یک بعدی را برداشته و آن را با مقدار بَعدی مقایسه می‌کنیم. اگر اولی از دومی بزرگتر باشد آن دو را جابه‌جا می‌کنیم و اگر بزرگتر نباشد کاری انجام نمی‌دهیم. سپس خانهٔ دوم را با سوم مقایسه می‌کنیم و همان کار پیشین را انجام می‌دهیم. هنگامی که به‌پایان آرایه برسیم، آخرین خانه، بزرگترین مقدار آرایه را در خود گرفته است (اگر نمی‌دانید چرا، می‌توانید با قلم و کاغذ روی چند عدد امتحان کنید).

حال دوباره از اول این کار را تکرار می‌کنیم، ولی این بار با رسیدن به‌خانهٔ پیش از آخری کار را متوقف می‌نمائیم. اکنون مقدار این خانه از همهٔ خانه‌های پیش از خود بزرگتر است. باز از اوّل کار را دوباره انجام می‌دهیم، و این قدر این کار را تکرار می‌کنیم تا همهٔ خانه‌های آرایه از کوچک به بزرگ آراسته و مرتب شوند. این نوع آرایش را، آرایش فرازی* (Ascending) می‌نامیم. در نمونهٔ زیر این روش را به‌کار بسته‌ایم. P آرایه‌ای است که می‌خواهیم آراسته شود. متغیّر Q تعدا خانه‌هایی را که می‌خواهیم مرتب کنیم نشان می‌دهد (چنانچه بخواهیم همهٔ خانه‌ها را آراسته کنیم مقدار آن برابر اندازهٔ آرایه خواهد بود):

```
200 FOR J = 1 TO Q - 1
210 F = 0
220 FOR K = 1 TO Q - J
230 IF P(K) = P(K + 1) THEN 280
240 T = P(K)
250 P(K) = P(K + 1)
260 P(K + 1) = T
270 F = 1
280 NEXT K
290 IF F = 0 THEN 310
```

```
300 NEXT J
310 . . .
.
.
.
```

متغیّر F برای این به‌کار رفته است که نشان دهد که آیا در هنگام پیمودن آرایه، جابه‌جایی خانه‌ها انجام شده است یا نه. اگر در سطر ۲۳۰ معلوم شود که خانه P(K) از خانهٔ بعدی، یعنی P(K+1)، بزرگتر است، در این صورت سطرهای ۲۴۰ تا ۲۷۰ اجراءِ می‌شوند وگرنه نمی‌شوند. متغیر T را به‌کار برده‌ایم تا بتوانیم مقدارهای دو خانه را جابه‌جا کنیم. سپس به متغیر F ، برای نشان دادن این که جابه‌جایی انجام پذیرفته است، مقدار ۱ را واگذار کرده‌ایم. هرگاه که پیمایش آرایه به پایان رسیده است، متغیّر F را در سطر ۲۹۰ وارسی کرده‌ایم. اگر متغیر F برابر صفر باشد (یعنی جابه‌جایی انجام نشده باشد)، آرایش انجام شده است و می‌توان کار را پایان بخشید. برای آزمایش این روش برنامهٔ زیر را نوشته و اجراءِ می‌کنیم:

```
100 DIM P(11)
110 DATA 27,14,6,8,2,19,19,20,16,1
120 FOR J = 1 TO 10
130 READ P(J)
140 NEXT J
150 PRINT
160 Q = 10


310 FOR J = 1 TO 10
320 PRINT P(J); PRINT" ";
340 PRINT
999 END
```

```
RUN

1 2 6 8 14 16 19 19 20 27
```

بخش هشتم

زیر روال ها

گاهی لازم است که در برنامه، بخشی از آن که کار خاصّی را انجام می‌دهد، چندین بار تکرار شود. مثلاً در یک برنامهٔ گزارش گیری ممکن است در چند جای برنامه بخواهیم جمع اعداد چاپ شده در یک صفحه را، در پایان صفحه چاپ کنیم، این جمع را به جمع کل بیافزائیم، سپس جمع صفحه را صفر کرده و به شماره صفحه یکی بیافزائیم. این کار را می‌توانیم با تکرار عبارت‌های موردنیاز، انجام دهیم:

```
340 PRINT "PAGE TOTAL =
350 G = G + P
360 P = 0
370 N = N + 1
.
.
.
510 PRINT "PAGE TOTAL = ";P
520 G = G + P
530 P = 0
540 N = N + 1
.
.
.
660 PRINT "PAGE TOTAL = ";P
670 G = G + 0
680 P = 0
690 N = N + 1
```

روشی دیگر که می‌توان به جای تکرار عبارت‌ها به کار برد، استفاده از «زیرروال*» (subroutine) می‌باشد. زیر روال بخشی از برنامه است، که معمولاً برای انجام کاری خاصّ نوشته شده است، و این بخش از برنامه را ممکن است از جای دیگر در برنامه به اجراءِ گذاشت.

دستور «رفتن به زیر روال» (GOSUB) کنترل برنامه را به شمارهٔ سطر داده شده منتقل می‌نماید:

```
200 GOSUB 1000
```

هنگام اجرای این دستور، دستگاه جای آن را بیاد می‌سپارد تا بتواند پس از انجام زیر روال، به‌عبارت بعد از این دستور بازگشت نماید. اکنون مثال قبلی را با استفاده از روش «زیر روال» انجام می‌دهیم.

```
.
.
.
340 GOSUB 900
350 . . .
.
.
510 GOSUB 900
520 . . . . . .
.
.
660 GOSUB 900
670 . . .
.
.
900 PRINT "PAGE TOTAL = ";P
910 G = G + P
920 P = 0
930 N = N + 1
940 RETURN
```

پس از اجرای دستور «رفتن به‌زیر روال» (مثلاً در سطر ۳٤۰)، کنترل برنامه به‌سطر ۹۰۰ منتقل می‌شود. سپس عبارت‌های ۹۰۰ تا ۹٤۰

انجام می‌شوند. دستور بازگشت (RETURN) کنترل را به‌سطر بعد از ۳۴۰ منتقل می‌نماید. دوباره هنگامی که دستور «رفتن به زیر روال» در سطر ۵۱۰ انجام شود، همان دستور بازگشت، کنترل را به‌سطر ۵۲۰ منتقل خواهد کرد.

در این جا بخشی را از یک برنامهٔ بازی می‌آوریم که در آن میان دستگاه و بازی کن گفت‌وگویی به‌عمل می‌آید:

```
800 IF W = 1 THEN 830
810 PRINT "HARD LUCK, I WIN"
820 GO TO 840
830 PRINT "YOU WIN, WELL DONE"
840 INPUT "PLAY AGAIN": A$
850 RETURN
.
.
```

هرگاه که یک دور بازی به‌پایان می‌رسد برنامه می‌تواند پیغام خود را فرستاده و بپرسد که آیا یک دور دیگر بازی خواهد شد یا نه. برای این منظور دستور زیر را اجراءِ می‌کند:

```
110 GOSUB 800
```

باید دقت به‌عمل آورد که به زیر روال جز به وسیلهٔ دستور «رفتن به‌زیر وال» وارد نشد، زیرا در این صورت پیغامی برای یادآوری این اشتباه فرستاده خواهد شد. برای نمونه، در برنامهٔ زیر اگر بازی کن پاسخی جز « YES » و «NO» بدهد، پیغام زیر فرستاده خواهد شد:

```
RETURN WITHOUT GOSUB IN 650

100
.
.
```

```
.
200 GOSUB 600
.
.
.
240 GOSUB 600
.
.
.
290 GOSUB 600
.
.
.
330 GOSUB 600
.
.
.
570 INPUT "ANY MORE";A$
580 IF A$ = "YES" THEN 100
590 IF A$ = "NO" THEN 999
600 PRINT
610 PRINT "SCORE SO FAR IS ";S
620 PRINT
630 INPUT "NEXT";N
640 M = M + 1
650 RETURN
999 END
```

این یک اشتباه سادهٔ برنامه نویسی است که ممکن است در هنگام آزمایش برنامه، متوجه آن نشویم (زیرا این اشتباه هنگامی روشن می‌شود که پاسخی جز « YES » و «NO» به‌دستگاه داده شود). برای اصلاح این اشتباه می‌توان دستور زیر را به‌برنامه افزود:

```
595 GO TO 570
```

شاید بهتر باشد که همهٔ زیر روال‌ها را در پایان برنامه جای داد، هرچند اجباری در انجام این کار نیست. باید پیش از هر «زیرروال» دستور جهش (GOTO) را به کاربرد تا از لغزیدن به داخل آن جلوگیری نماید.

گاهی ممکن است بخواهیم در داخل زیر روال دستور خاصی را انجام دهیم که هر بار با مقدار متفاوتی سر و کار داشته باشد. برای مثال، اگر بخواهیم در جاهای مختلف برنامه نشانه‌های گوناگونی را به‌تعداد نابرابر چاپ کنیم می‌توانیم به‌روش زیر عمل نمائیم:

```
.
.
220 FOR J = 1 TO 6
230 PRINT "*";
240 NEXT J
250 PRINT
.
.
.
300 FOR K = 1 TO 10
310 PRINT "-";
320 NEXT K
330 PRINT
.
.
.
460 FOR J = 1 TO 5
470 PRINT ".";
480 NEXT J
490 PRINT
```

زیرروال زیر را، که در آن از متغیرهای جانشینی استفاده شده است، می‌توان به‌جای بخش‌های تکراری بالا به‌کار برد:

```
600 FOR J = 1 TO L
610 PRINT L$;
620 NEXT J
630 PRINT
640 RETURN
```

پیش از دستور «رفتن به‌زیرروال» باید متغیرها را با مقدارهای مورد نظر پر کرد:

```
220 L = 6
230 L$ = "*"
240 GOSUB 600
.
.
.
300 L = 10
310 L$ = "-"
320 GOSUB 600
.
.
.
460 L = 5
470 L$ = "."
480 GOSUB 600
.
.
```

متغیرهای L و *L$* راپارامترهای زیرروال می‌نامیم. هر چند در این مثال از تعداد سطرهای برنامه کاسته نشده است ولی در برنامه‌های واقعی، با به‌کار بردن این روش، معمولاً از تعداد سطرها بسیار کاسته می‌شود.

پارامترها را می‌توان هم برای فرستادن و هم برای گرفتن اطلاعات به‌کار برد. زیرروال زیر که برای «درست‌سازی» داده‌ها نوشته شده است این کاربرد را نشان می‌دهد:

```
200 INPUT "WHICH MONTH";M
210 GOSUB 400
220 IF F = 1 THEN 200


400 F = 1
410 IF M > 12 THEN 440
430 IF M < 1 THEN 440
430 F = 0
440 RETURN
```

پارامتر M شماره ماه را به‌زیرروال می‌برد و متغیر F مقداری‌را از زیرروال برمی‌گرداند که نشان دهندهٔ درستی یا نادرستی شمارهٔ ماه می‌باشد.

در عمل، چنین زیرروالی را می‌توان به‌صورت جامعتر مورد استفاده قرار داد. در زیرروال زیر وارسی کاملی از درستی تاریخ، که آنرا در متغیّرهای D و M و Y (به ترتیب برای روز و ماه و سال) قرار داده‌ایم، به‌عمل می‌آید:

```
800 F = 1
810 IF M > 12 THEN 960
820 IF M < 1 THEN 960
830 IF D < 1 THEN 960
840 IF D > 31 THEN 960
850 IF M <> 2 THEN 890
860 IF D > 29 THEN 960
870 IF Y - (INT(Y/4)*4) = 0 THEN 950
880 IF D > 28 THEN 960
885 GO TO 950
890 IF D < 31 THEN 950
900 IF M = 2 THEN 960
910 IF M = 4 THEN 960
920 IF M = 6 THEN 960
930 IF M = 9 THEN 960
940 IF M = 11 THEN 960
950 F = 0
960 RETURN
```

(مفهوم سطر ۸۷۰ را، که برای وارسی سال کبیسه به کار رفته است، در بخش نهم شرح خواهیم داد.)

شما می‌توانید منطق زیرروال بالا را، با استفاده از مداد و کاغذ، و به کار بردن تاریخ‌های گوناگون درست و نادرست، وارسی نمائید. چنانچه تاریخ‌ها مربوط به‌سدهٔ بیستم باشند، می‌توان دو عبارت زیر را نیز افزود:

```
803 IF Y < 1900 THEN 960
807 IF Y > 1999 THEN 960
```

استفاده از زیرروال در برنامه به‌دو دلیل سودمند است. نخست، چنان که پیش از این دیدیم، با جلوگیری از تکرار عبارت‌ها اندازه برنامه را کاهش می‌دهد. دوم این که این کار به گسترش کتابخانهٔ زیرروال‌ها می انجامد.

هنگامی که یک زیرروال نوشته و آزمایش شد، می‌تواند آن را برای استفادهٔ دوباره نگهداری کرد. زیرروال‌ها را می‌توان یا به‌سادگی بر روی کاغذ نگهداری کرد، یا از نوار و دیسکت برای این کار استفاده کرد. در هنگام تهیه زیرروال بهتر است که شماره سطرها را از اعداد بزرگ انتخاب کرده (مثلاً از ۷۰۰۰ به بالا) و شمارهٔ سطرهای آغاز و پایان آن‌را، برای مراجعه بعدی، در لیستی یادداشت نمائیم.

این لیست می‌تواند به شکل زیر باشد:

وارسی تاریخ:	۷۰۰۰ تا ۷۰۶۰
پاک کردن صفحهٔ نمایش:	۷۱۰۰— ۷۱۷۰
صدای سوت دستگاه:	۷۲۰۰— ۷۲۵۰
چاپ جمع‌ها:	۷۳۰۰— ۷۵۱۰
وارسی صفحه کلید:	۷۵۵۰— ۷۶۱۰
بستن پرونده‌ها:	۷۶۵۰— ۷۷۶۰
محاسبهٔ ریشهٔ دوم:	۷۸۰۰— ۷۹۱۰

آخرین زیرروال بالا، یعنی محاسبهٔ ریشه دوم، می‌تواند به گونهٔ زیر نوشته شود. عددی که می‌خواهیم ریشه دوم آن را به‌دست آوریم در S و ریشهٔ دوم آن در R قرار خواهند گرفت:

```
7800 L = 0
7810 H = 182
7820 I = (L + H)/2
7830 J = S/I
7840 IF J = I THEN 7910
7850 IF H - L < 2 THEN 7910
7860 IF J < I THEN 7890
7870 L = I
7880 GO TO 7820
7890 H = I
7900 GO TO 7820
7910 R = (I + J)/2
7920 RETURN
```

زیرروال‌های تودرتو

یک زیرروال می‌تواند زیرروال دیگری را فرا خوانده و مورد استفاده قرار دهد. برای نمونه، زیرروال زیر که برای دریافت کردن داده‌ها به‌کار رفته است، خود زیرروال وارسی تاریخ را، که از سطر ۷۰۰ شروع می‌شود، به‌کار برده است:

```
1000 PRINT
1010 INPUT "NAME";N$
1020 INPUT "DATE OF BIRTH,DD,MM,YY";D,M,Y
1030 GOSUB 7000
1040 IF F = 0 THEN 1070
1050 PRINT "DATE INVALID, TRY AGAIN"
1060 GOTO 1020
1070 INPUT "SEX";S$
1080 IF S$ = "M" THEN 1120
1090 IF S$ = "F" THEN 1120
1100 PRINT "SEX INVALID,ENTER M OR F"
1110 GOTO 1070
1120 RETURN
```

به‌شیوهٔ فراخوانی یک زیرروال از درون یک زیرروال دیگر روش «تودرتوسازی» گفته می‌شود. معمولاً تا ۲۵ زیرروال را می‌توان به‌ترتیب از داخل هم فرا خواند.

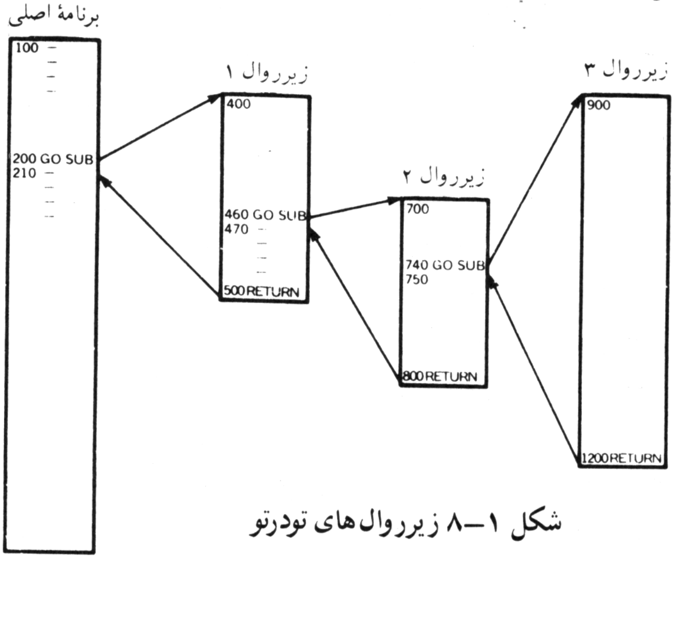
این نکته گفتنی است که ترتیب اجرای دستورهای بازگشت، برعکس ترتیب اجرای دستورهای «رفتن به‌زیرروال» است. یعنی از زیر روالی که زودتر وارد شده‌ایم دیرتر خارج می‌شویم. بیسیک برای به‌یاد سپردن محل بازگشت روش خاصی را به‌کار می‌برد. هنگامی که دستور «رفتن به‌زیرروال» اجراءِ می‌شود، بیسیک شماره سطر بعد از این دستور را برداشته و آن را در آخرین خانهٔ آرایه‌ای که مخصوص این کار است قرار می‌دهد.

اگر دوباره به‌دستور «رفتن به‌زیرروال» برسد، باز آدرس محل بازگشت را در همان آرایه، ولی این بار درخانهٔ پیش از آخر قرار می‌دهد.

این کار برای هر فرا خواندن زیرروال تکرار می‌شود. آرایه را «پشته*» و این روش را، که شماره‌ها روی هم چیده می‌شوند، «پشته‌سازی*» می‌نامیم. برعکس هنگام اجرای یک دستور بازگشت، نشان محل بازگشت از بالای پشته برداشته شده و همزمان این شماره از پشته حذف می‌شود.

این روش دو مشکل را به‌همراه می‌آورد: نخست این که اگر دستور بازگشت پیش از دستور فرا خواندن زیرروال انجام شود، چون پشته هنوز خالی است، این کار اشتباه به‌حساب می‌آید. دیگر این که دستورهای بازگشت باید درست برعکس دستورهای «فراخوانی» انجام شوند تا منطق برنامه درست اجراءِ شود.

شکل ۱ــ۸ نشان دهندهٔ سه زیرروال تودرتو است. هنگامی که زیرروال سوم اجراءِ می‌شود پشتهٔ شماره سطرها به ترتیب دارای شماره‌های ۷۵۰، ۴۷۰ و ۲۱۰ می‌باشد. حال اگر دستور بازگشت انجام شود، حتی اگر این بازگشت از داخل زیرروال سوم نباشد، کنترل برنامه به‌زیرروال دوم منتقل خواهد شد.



شکل ۱ــ۸ زیرروال‌های تودرتو

در بعضی از گونه‌های بیسیک، زیرروال می‌تواند خودش را فرا خواند. این روش «خود فراخوانی*» نامیده می‌شود و از آن در پردازش‌های تکراری، برای حل مسایل ریاضی استفاده می‌شود. برای مثال، زیرروال حود فرا خوان زیر را به کار برده‌ایم تا با روش اقلیدسی بزرگترین عامل مشترک دو عدد را به‌دست آوریم. این عددها در پارامترهای H و L و نتیجه به‌دست آمده در متغیر F قرار می‌گیرند. توجه کنید که تابع عدد صحیح (INT) برای برداشتن قسمت صحیح یک عدد اعشاری به کار می‌رود (این تابع را به‌تفصیل در بخش نهم شرح خواهیم داد):

```
8000 F = H
8010 L = H - L*INT(H/L)
8020 H = F
8030 IF L = 0 THEN 8050
8040 GOSUB 8000
8050 RETURN
```

برای این که طرز کار این زیرروال را دریابید، فرض کنید که زیرروال را به‌شیوهٔ زیر فرا خوانیم:

```
100 H = 204
110 L = 85
120 GOSUB 8000
130 . . .
```

برای هر تکرار زیرروال، «پشته» و متغیرها به‌ترتیب دارای مقدارهای زیر می‌باشند:

	F = 204 85 34 17
8050	L = 85 34 17 0
8050	H = 204 85 34 17
8050	
130	
پشته	**محتویات پشته**

دستور «فرا خواندن»

این دستور برای فرا خواندن روال (Routine) هایی که به‌زبان ماشین نوشته شده است به کار می‌رود. این کار به‌دو دلیل انجام می‌شود.

سیستم‌های عامل نه به‌زبان بیسیک، بلکه به‌زبانی نزدیک به‌زبان ماشین نوشته می‌شوند، هر چند شاید لازم نباشد که برنامه‌نویس خود را گرفتار این مسئله نماید، ولی معمولاً سیستم‌های عامل دارای روال‌های سودمندی هستند که می‌توان آن‌ها را به‌وسیلهٔ بیسیک فرا خواند. مثلاً دستگاه اَپل با دستور زیر صفحه نمایش را پاک کرده و مکان‌نما را به‌خانه (گوشه بالای دست چپ) می‌برد:

```
100 CALL 64600
```

با این دستور کنترل به‌محل ٦٤٦٠٠ حافظه منتقل می‌شود. پس از انجام روال یاد شده، کنترل به‌سطر بعدی منتقل خواهد شد. دستگاههای مختلف، برای استفاده از تابع‌های سیستم عامل، روال‌های مختلفی را پیش‌بینی کرده‌اند. این روال‌ها در کتاب‌های مربوط به‌دستگاه، به تفصیل شرح داده شده‌اند.

دستور فرا خواندن را می‌توان همچنین برای دستیابی بر زیرروال‌هایی که خود به‌زبان ماشین نوشته‌ایم به کار برد. معمولاً، این زیرروال راها را برای پردازش‌هایی که باید بسیار سریع انجام شوند (مثلاً در بازی های کامپیوتری) به‌کار می‌بریم. برای نوشتن روال‌ها به‌زبان ماشین، باید از دستور «بایت گذاری» (POKE) استفاده کرد.

```
100 POKE 27,40120
110 POKE 34,40121
120 POKE 66,40122
```

در این جا بیش از این به‌زبان ماشین نمی‌پردازیم. خواننده می‌تواند به کتاب‌های مربوط به‌آن مراجعه نماید. از دستور بالا همچنین می‌توان برای تغییر مقدارهایی در سیستم عامل استفاده کرد. مثلاً در دستگاه اَپل، با دستور

زیر می‌توان تعداد نشانه‌های هر سطر را، بر روی صفحهٔ نمایش، به ۲۰ تغییر داد:

```
100 POKE 33,20
```

مکمّل دستور بایت گذاری دستور «بایت برداری» (PEEK) است که به‌وسیلهٔ آن برنامه‌نویس می‌تواند اطلاعاتی را از داخل‌سیستم عامل بردارد. برای مثال، در همان دستگاه اَپل، برای این که محل افقی مکان‌نما را به‌دست آوریم، دستور زیر را به کار می‌بریم:

```
110 H = PEEK(36)
```

برای اطلاع کامل از آن چه در هر دستگاه می‌توان با استفاده از دو دستور بالا به‌سیستم عامل داد یا از آن گرفت، خواننده باید به کتاب‌های راهنمای دستگاه مراجعه نماید.

بخش نهم
تابع ها

از تابع ها برای انجام عملیات ریاضی که با به کار بردن نشانه های +، —، *، / و ↑ شدنی نیستند استفاده می‌کنیم؛ و نیز چنان که در بخش هفتم شرح دادیم، تابع ها را برای انجام کارهایی بر روی رشته های حرفی به کار می‌بریم.

تابع مقدار صحیح (INT) را که پیش از این در بعضی مثال ها به کار بردیم، می‌توان به‌عنوان نمونهٔ این ویژگی بیسیک به‌شمار آورد. از این تابع برای جدا ساختن مقدار صحیح یک عدد اعشاری استفاده می‌شود. برای مثال، عبارت زیر نخست متغیر A را بر ۲ بخش کرده، سپس مقدار صحیح آن را برداشته و به‌متغیّر R واگذار می‌کند:

```
100 R = INT(A/2)
```

بنابراین مقدار R عدد صحیحی خواهد بود که کوچکتر یا مساوی خارج قسمتی است که از تقسیم A بر ۲ به‌دست می‌آید. اگر A مساوی ۷ باشد خارج قسمت به‌دست آمده ۳/۵ می‌شود، که مقدار صحیح آن یعنی ۳ به R داده می‌شود. در زیر نمونه های بیشتری را آورده‌ایم:

ارزش یک عبارت	مقدار صحیح آن
۲/۹	۲
۳/۱	۳
٦/۱	٦
٦/۹۹۹۹	٦
٦/۰۰۰۱	٦
۲٦/۲	۲٦

گفتیم که تابع مقدار صحیح، مقداری را که از یک عدد اعشاری برمی‌دارد کوچکتر یا برابر آن عدد است. از این‌رو اگر عدد اعشاری منفی باشد، مقدار صحیح آن از نظر قدر مطلق ممکن است با آن برابر یا از آن بزرگتر باشد (مثلاً: INT(−3.1)= −4.)

از تابع مقدار صحیح می‌توان برای گرد کردن عددها (یعنی تبدیل عدد اعشاری به‌عدد صحیحی که به آن نزدیک است) استفاده کرد:

```
110 R = INT (V + 0.5)
```

در برنامهٔ زیر این تابع را به‌کار برده‌ایم تا تعداد قوطی های رنگی را که برای رنگ کردن دیواره‌ها و سقف یک بشکهٔ آب لازم است محاسبه نمائیم:

```
110 INPUT "TANK DIAMETER";D
120 INPUT "TANK HEIGHT";H
130 INPUT "COVERAGE PER POT OF PAINT"; C
140 N = INT (3.14*D*(D/4 + H))/C
150 PRINT
160 PRINT N + 1;" TINS OF PAINT REQUIRED"
170 INPUT "MORE";Y$
180 IF Y$ = "YES" THEN 110
999 END
```

تابع ریشهٔ دوم (SQR) برای به‌دست آوردن ریشهٔ دوّم متغیرها به کار

می‌رود:

210 R = SQR(A)

با این دستور ریشهٔ دوم A به‌دست آمده و به‌متغیر R واگذار می‌شود.

از تابع‌ها، در هر جایی که بتوان یک متغیر را به‌کار برد، می‌توان استفاده کرد؛ البته باید عددی یا حرفی بودن متغیرها را در نظر داشت (زیرا بعضی تابع‌ها با متغیرهای عددی و برخی با متغیرهای حرفی کار می‌کنند). از این رو می‌توان با دستور زیر، ریشه چهارم متغیر F را به‌دست آورد:

220 R = SQR(SQR(F))

در بعضی از گونه‌های بیسیک اگر مقداری که با تابع ریشه دوم به‌کار می‌رود منفی باشد، برنامه از کار باز می‌ماند ولی در بعضی دیگر، برنامه متوقف نمی‌شود ولی نتیجهٔ به‌دست آمده نادرست می‌باشد. برای اینکه مطمئن شویم که مقدار مورد نظر همواره مثبت است، از تابع قدر مطلق استفاده می‌کنیم.

تابع قدر مطلق (ABS)، قدر مطلق عبارت‌ها و متغیرها را برمی‌دارد، یا به سخن دیگر، مقدار مثبت آن‌ها را به‌دست می‌دهد. از این رو اعداد ۲۷ و ۲۷− هر دو عدد ۲۷ را برمی‌گردانند. پس اگر X عددی بزرگتر از صفر باشد ABS(X) = X ؛ و اگر X از صفر کوچکتر باشد X− = ABS(X) .

عبارت بالا را می‌توانیم چنین بازنویسی کنیم:

220 R = SQR (SQR(ABS(F)))

گاهی لازم است بدانیم که مقدار یک متغیر یا یک عبارت مثبت است یا منفی. برای این منظور تابع علامت (SGN) را به‌کار می‌بریم. این تابع برای مقدارهای مثبت عدد ۱، برای مقدارهای منفی عدد ۱− و برای مقدارهای صفر عدد صفر را به‌دست می‌دهد.

در زبان بیسیک دو تابع لگاریتمی هم به‌کار می‌رود. برای محاسبهٔ لگاریتم طبیعی اعداد از تابع لگاریتم (LOG) استفاده می‌کنیم.

```
150 L = LOG(N)
```

برای این که تابع درست کارکند باید N بزرگتر از صفر باشد. می‌توان پیش از دستور بالا، علامت N را وارسی کرد.

```
140 IF SGN (N) <1 THEN 160
```

کافی نیست که عبارت زیر را به کار بریم، زیرا N ممکن است دارای مقدازی بزرگتر از صفر و کوچکتر از ۱ (مثلاً ۰/۲) باشد.

```
140 IF N <1 THEN 160
```

برای محاسبهٔ آنتی‌لگاریتم طبیعی یا توان‌های عدد نِپِر (یعنی مقدار e^x وقتی که ۲/۷۱۸۲۸= e باشد)، از تابع نمایی (EXP) استفاده می‌کنیم:

```
170 A = EXP(N)
```

در این مثال e به‌توان N رسیده و مقدار آن به‌متغیر A داده می‌شود.

تابع‌های مثلثاتی

برای محاسبهٔ سینوس، کسینوس، تانژانت و کمانِ تانژانت زاویه‌هایی که بر حسب رادیان باشند. تابع هایی وجود دارد. هر رادیان برابر $\frac{۱۸۰}{\pi}$ **درجه** می‌باشد. برای تبدیل درجه به رادیان از روش زیر استفاده می‌شود:

```
100 P = 3.1416
110 INPUT "ANGLE IN DEGREES";D
120 R = D*P/180
130 PRINT " EQUALS ";R;" RADIANS"
999 END
```

راه دیگر این است که از ضریب $\frac{\pi}{۱۸۰}$ استفاده کنیم:

```
100 F = 3.1416/180
120 R = D*F
```

در زیر چند زاویه را بر حسب رادیان و درجه می‌آوریم:

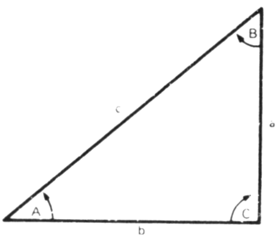
2π رادیان = ۳۶۰ درجه

π رادیان = ۱۸۰ درجه

۱ رادیان = $\frac{۱۸۰}{\pi}$ درجه

$\frac{\pi}{۱۸۰}$ رادیان = ۱ درجه

برای خوانندگانی که مثلثات دوران مدرسه را فراموش کرده‌اند، در شکل ۱—۹ نسبت‌های اصلی را نشان داده‌ایم:



cos A = b/c

sin A = a/c

tang A = a/b

شکل ۱—۹ نسبت‌های مثلثاتی

برای محاسبهٔ سینوس یک زاویه، تابع سینوس (SIN) را به کار می‌بریم:

```
100 S = SIN(A)
```

و برای محاسبهٔ کسینوس یک زاویه، تابع کسینوس (COS) را:

```
110 C = COS(A)
```

در محاسبهٔ تانژانت زاویه از تابع تانژانت (TAN) استفاده می‌کنیم:

```
120 T = TAN(A)
```

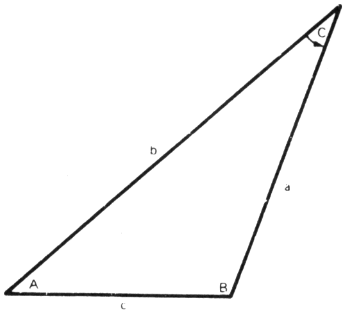
و در محاسبهٔ کمان تانژانت (یا زاویه‌ای که تانژانت آن را می‌دانیم) از تابع کمان تانژانت (ATN):

```
130 A = ATN(T)
```

در مثال زیر با استفاده از تابع‌های مثلثاتی برنامه‌ای نوشته‌ایم که مساحت مثلث و ضلع c آن را محاسبه می‌نماید. از این مثلث که در شکل

۲ـ۹ نشان داده شده است دو ضلع a و b و زاویهٔ بین آن‌ها C معلوم است. با استفاده از دو قاعدهٔ زیر می‌توان مساحت و ضلع دیگر مثلث را به‌دست آورد:

سطح Area = ½ ab sin C
ضلع Side c = $a^2b^2 - 2ab \cos C$



شکل ۲ـ۹ ضلع‌های a و b وزاویهٔ C معلوم است.

```
100 INPUT "LENGTHS OF SIDES";A,B
110 INPUT "INCLUDED ANGLE IN DEGREES";C
120 R = C* 3.1416/180
130 PRINT "AREA IS "; 0.5*A*B*SIN(R)
140 PRINT "THIRD SIDE LENGTH IS ";
150 PRINT SQR (A↑2 + B↑2 - 2*A*B*COS(R))
160 PRINT
170 INPUT "ANY MORE";Y$
180 IF Y$ = "YES" THEN 100
999 END
```

بسیاری از گونه‌های بیسیک تنها تابع‌هایی را که نام بردیم دارا می‌باشند. این تابع‌ها را «درونی» می‌نامیم. گفتیم که چهار تابع مثلثاتی داریم؛ مقدارهای مثلثاتی دیگر را می‌توان با استفاده از رابطه‌های موجود میان آن‌ها و چهار تابع یاد شده محاسبه کرد. مشهورترین رابطه‌های مثلثاتی در زیر نشان داده شده‌اند:

sec = 1/ cos

cosec = 1/ sin

catang = 1/ tang

برنامه‌نویس می‌تواند با استفاده از عبارت «تعریف تابع» (DEF) در هر برنامه تابع‌هایی را تعریف نماید. در هر برنامه می‌توان تا ۲۶ تابع (یعنی از FNA تا FNZ) معرفی کرد. در زیر تابعی را به‌نام FNS برای محاسبهٔ عکس سینوس (cosec) به کار برده‌ایم:

```
100 DEF FNS(A) = 1/SIN(A)
```

حرف A را که پس از FNS در داخل پرانتز قرار داده‌ایم «شناسهٔ ساختگی» می‌نامیم چون عملاً کاربردی ندارد. هرجا که از تابع FNS استفاده کنیم، بیسیک معادل آن را از دستور تعریف تابع (DEF) برداشته و در جای آن می‌نشاند. از این رو در برنامه‌ای که دستور بالا را به کار برده باشیم دو عبارت زیر هم ارز می‌باشند:

```
100 S = FNS(A)
110 S = 1/SIN(A)
```

تابع زیر را برای تبدیل درجه به رادیان تعریف کرده‌ایم، تا در برنامه‌های مثلثاتی از آن استفاده کنیم:

```
100 DEF FNR(A) = A*3.1416/180
```

سپس می‌توانیم به‌روش زیر آن را به کار بریم:

```
220 INPUT "ANGLE IN DEGREES";C
230 R = FNR(C)
```

در مثال زیر نشان داده‌ایم که تابع‌های تعریف شده می‌توانند تابع‌های «درونی» را در خود داشته باشند. این تابع برای محاسبهٔ «کمان سینوس» (arcsin) یک زاویه تعریف شده است.

```
130 DEF FNT(X) = ATN(X/SQR(- X*X + 1))
```

چنانچه بخواهیم عبارت پیچیده بالا را بارها در یک برنامه به‌کار ببریم، بسیار خردمندانه است که آن را به‌صورت یک تابع معرفی نمائیم.

در تعریف تابع‌ها می‌توان از متغیّرهای حرفی نیز استفاده کرد. در مثال زیر تابعی معرفی شده است که حرف میانی یک رشتهٔ حرفی را (که تعداد حرف‌های آن فرداست) برمی‌دارد:

```
100 DEF FNA(A$) = MID$(A$,ABS(LEN(A$) + 0.5),1)
```

عددهای تصادفی

بیسیک دارای تابع درونی دیگری نیز می‌باشد که در نوشتن برنامه‌های بازی بسیار کاربرد دارد. این تابع که تابع «تصادفی» (RND) نام دارد برای به‌دست آوردن عددی تصادفی میان صفر و ۱ به‌کار می‌رود. این تابع نیز دارای شناسه‌ای (Argument) می‌باشد که در دستگاههای گوناگون کاربرد کم و بیش متفاوتی دارد.

اگر این شناسه بزرگتر از صفر باشد، هرگاه این تابع به‌کار رود، عددی تازه‌ای به‌دست خواهد آمد:

```
120 R = RND(1)
```

اگر شناسه صفر باشد، نتیجهٔ به‌دست آمده از تابع قبلی تکرار می‌شود. از این‌رو اگر عبارت ۱۳۰، درست بعد از عبارت بالا به‌کار رود S و R دارای مقداری مساوی خواهند بود:

```
130 S = RND(0)
```

چنانچه شناسه عددی بزرگتر از صفر باشد هرگاه که برنامه اجراءِ شود، عددهای تصادفی به‌دست آمده همان عددهای اجرای پیشین خواهند بود. این عددها را می‌توان با تغییر شناسه تغییر داد. اگر شناسه مقدار منفی خاصّی باشد، عدد به‌دست آمده همواره یکسان خواهد بود.

```
110 T = RND(-3)
```

این تکراری بودن عدد برای پیدا کردن اشکالات برنامه مفید است.

رشتهٔ عددهای تصادفی که به وسیلهٔ دستور عدد تصادفی به‌دست می‌آیند، در حقیقت عددهای «تصادفی‌نما» هستند؛ زیرا عددهای تصادفی واقعی همواره و در هر اجراءِ باید با عددهای قبلی تفاوت داشته باشند. برای برطرف کردن این اشکال به‌دو روش می‌توان عمل کرد: نخست این که از استفاده کننده بخواهیم عددی را به‌دستگاه بدهد:

```
100 INPUT "START OF PROCESSING" ; S
110 D = RND(-S)
```

راه دوم، به‌کار بردن حلقه‌ای است که تا هنگامی که یک عمل خارجی (مثلاً فشار دادن یکی از دکمه‌های صفحه کلید) انجام می‌شود تکرار می‌گردد، و همراه با تکرار مقدار شمارنده‌ای را افزایش می‌دهد. این شمارنده را می‌توان به‌عنوان شناسه تابع عدد تصادفی به‌کار برد. عددی را که به‌وسیلهٔ تابع عدد تصادفی به‌دست می‌آید، می‌توان به سادگی به‌عددی که در دامنهٔ خاصی قرار داشته باشد (مثلاً میان ۱ و ۱۰) تبدیل کرد. در مثال زیر این دامنه را از ۱ تا ۶ گرفته‌ایم:

```
D = INT (RND(R)*6) + 1
```

هرچند که عدد تصادفی میان صفر تا ۱ خواهد بود، ولی D مقداری میان ۱ تا ۶ خواهد داشت.

برخی از گونه‌های بیسیک دستوری را به‌کار می‌برند که به‌وسیلهٔ آن ترتیب اعداد تصادفی به‌هم میخورد:

```
100 RANDOMISE
```

با مراجعه به‌کتاب راهنمای دستگاه، می‌توان دریافت که آیا این دستور در آن قابل اجرا هست یا نیست.

بخش دهم

نگاره‌سازی

ریز کامپیوترها برای کشیدن تصویرها روش نگاره‌سازی را به کار می‌برند. این روش به خصوص برای تصویری کردن بازی‌ها— همچون: «مهاجمان فضایی»، «انفجار» و شطرنج— بسیار سودمند است.

هرچند اهمیّت نگاره‌سازی در کارهای بازرگانی بدین روشنی نیست، ولی در آن جا هم می‌توان آن را همچون ابزاری نیرومند به کار برد. با توجه به این گفتهٔ هنرمندان که: «یک تصویر به هزار کلمه می‌ارزد»، روشن می‌شود که اطلاعات بازرگانی را، اگر به وسیلهٔ نمودارها و نگاره‌ها نشان داده شوند، می‌توان بسیار زودتر دریافت.

برای کشیدن یک تصویر، صفحهٔ نمایش را به تعداد زیادی مربع کوچک تقسیم کرده و سپس با سیاه کرده برخی از آن‌ها شکل موردنظر را تهیّه می‌نمایند. (با سیاه کردن تمام صفحه و سفید نشان دادن بعضی از مربع‌ها، می‌توان تصویر «منفی» یک شکل را به دست آورد. هرچند برای آسانی کار، ما همهٔ نمونه‌ها را از تصویرهای «مثبت» انتخاب کرده‌ایم.) این مربع‌های کوچک را «یاختهٔ تصویری» (pixel) می‌نامیم.

یاخته‌های تصویری در ساده‌ترین شکل خود با جایی که یک نشانه در روی صفحهٔ نمایش می‌گیرد یکسان می‌باشند. این یاخته‌ها به جای این که فاصله یا نشانه‌ای را نشان دهند، مربع سیاه یا سفیدی را نمایش می‌دهند. یاخته‌های تصویری را، برحسب جایشان در سطر و ستون، شماره‌گذاری

کرده‌اند. از این رو در صفحه نمایشی که از ۲۴ سطر و ۴۰ ستون ساخته شده باشد، ۹۶۰ یاخته وجود دارد. هریک از این یاخته‌ها را می‌توان با دستور «نگاره‌گذاری» (PLOT) سیاه کرد:

```
100 PLOT 20,11
```

عبارت بالا یاخته‌ای را که در سطر ۱۲ و ستون ۲۰ قرار دارد سیاه می‌کند (سطر و ستون از شماره صفر به بالا شماره‌گذاری شده‌اند). میتوان محورهای مختصاتی در نظر گرفت که x های آن را شماره ستون‌ها و y های آن را شمارهٔ سطرها تشکیل داده باشند.

```
210 PLOT X,Y
```

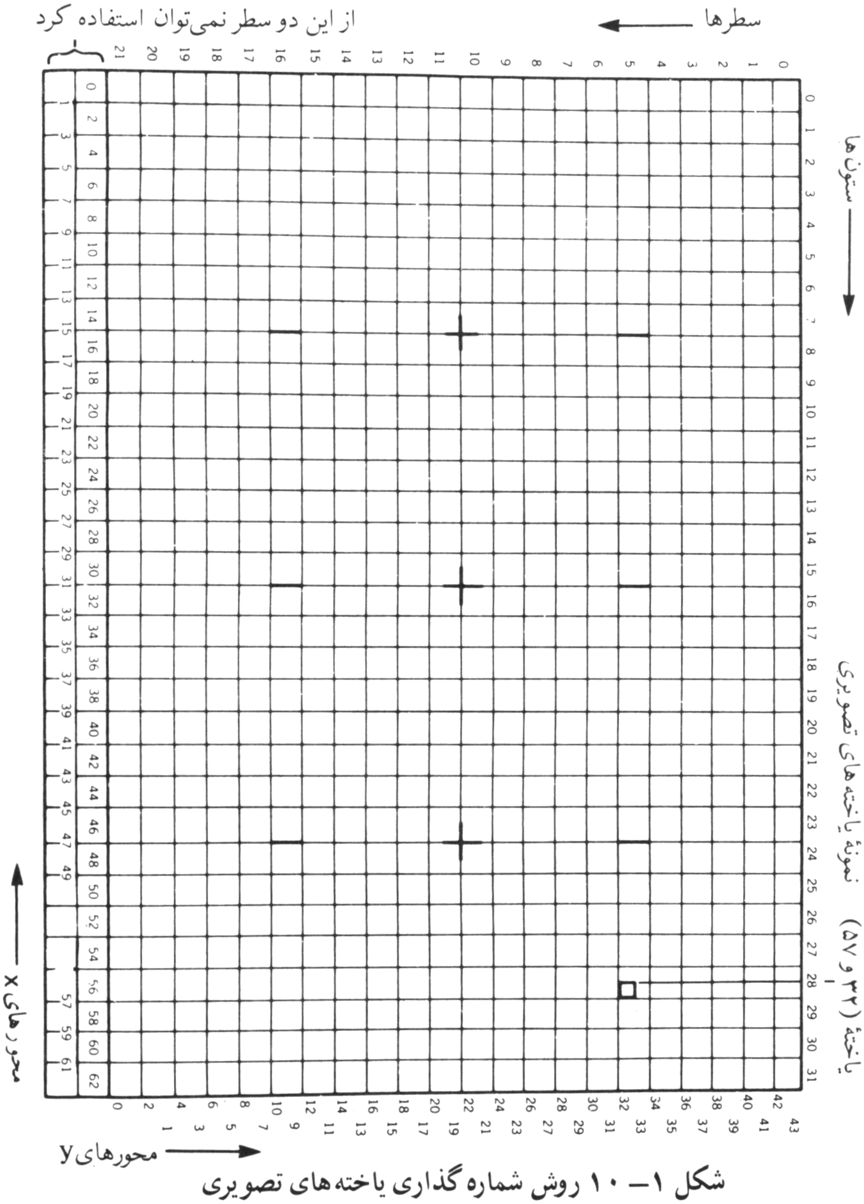
هر متغیر، ثابت یا عبارت ریاضی را که مقدار آن در محدودهٔ درست قرار گیرد، می‌توان به جای X و Y دستور بالا به کار برد. بعضی از دستگاه‌ها در جای هر نشانه، چهار یاخته مساوی را نشان می‌دهند. با این کار تعداد یاخته‌های تصویری بیشتر می‌شود. مثلاً اگر صفحهٔ نمایش از ۲۲ سطر و ۲۳ ستون ساخته شده باشد. تعداد یاخته‌های تصویری آن ۲۸۱۶ عدد می‌باشد. این افزایش یاخته‌ها دقیقتر شدن تصویر را به دنبال دارند برای سیاه کردن این یاخته‌ها نیز از همان دستور قبلی استفاده می‌شود:

```
230 PLOT 60,30
```

روش پاک کردن یاخته‌های تصویری سیاه شده، به دستگاه مورد استفاده بستگی دارد. اگر دستگاه تنها سیاه و سفید باشد می‌توان دستور «نگاره‌زدایی» (UNPLOT) را به کار برد.

```
300 UNPLOT X,Y
```

شکل ۱— ۱۰ نشان دهندهٔ طرح‌بندی صفحه‌ای است که هم‌اکنون شرح دادیم. مربع‌های بزرگتر جای چاپ نشانه‌ها و مربع‌های کوچکتر، یاخته‌های تصویری را نشان می‌دهند. در بسیاری از دستگاه‌ها، دو سطر پایانی صفحه را نمی‌توان در نگاره‌سازی به کار برد، زیرا این دو سطر برای نمایش



شکل ۱ــ ۱۰ روش شماره گذاری یاخته‌های تصویری

فرمان‌های دستگاه پیش‌بینی شده‌اند. در دستگاههایی که دارای صفحهٔ نمایش رنگی هستند، باید پیش از دستور نگاره‌گذاری با دستور «رنگ‌آمیزی» (COLOR) رنگ نگاره را مشخص نمائیم. برای مثال، فرض کنید بخواهیم مربع سیاهی را نمایش داده و سپس آن را از روی صفحه، که رنگ آن سفید است، حذف نمائیم. شمارهٔ رنگ‌های سفید و سیاه به ترتیب ۱۵ و صفر می‌باشد:

```
200 REM PLOT A BLACK SQUARE
210 COLOR = 0
220 PLOT 20,10
.
.
.
.
400 COLOR = 15
410 PLOT 20,10
.
.
```

(توجه کنید که واژهٔ رنگ (COLOR) با املای آمریکایی نوشته می‌شود.)

همین گونه، در برخی از دستگاهها، همچون اَپِل، لازم است که صفحهٔ نمایش را برای چاپ حروف (TEXT) یا نگاره‌سازی (GR)، با دستورهای مناسب آماده ساخت:

```
210 TEXT
.
.
.
350 GR
```

دستگاههای گوناگون، برای نگاره‌سازی دستورهای مختلفی را به کار می‌برند. نمونهٔ دستورهایی که در بعضی از دستگاهها قابل استفاده است، در زیر نشان داده می‌شود:

```
100 HLIN 1,6 AT 5
```

دستور بالا میان یاخته‌های یکم تا ششم سطر پنجم، خطّی افقی رسم می‌کند. به زبان دیگر، این دستور با دستورهای نگاره‌گذاری زیر، هم ارز می‌باشد:

```
100 PLOT 1,5
110 PLOT 2,5
120 PLOT 3,5
130 PLOT 4,5
140 PLOT 5,5
150 PLOT 6,5
```

به همین گونه، دستور زیر برای رسم خط عمودی به کار می‌رود:

```
100 VLIN 2,10 AT 8
```

این دو دستور را می‌توان برای کشیدن مربع در کنارهٔ صفحهٔ نمایش به کار برد: اگر صفحه دارای ۲۴×۴۰ یاختهٔ تصویری باشد، دستورهای زیر را باید به دستگاه داد:

```
100 HLIN 0,39 AT 0
110 HLIN 0,39 AT 23
120 VLIN 0,23 AT 0
130 VLIN 0,23 AT 39
```

برای این که رنگ هر یاخته را به دست آوریم، دستور زیر را به کار می‌بریم:

```
100 C = SCRN (10,14)
```

پاک کردن صفحهٔ نمایش

برای پاک کردن صفحه نمایش چنین دستورهایی را به کار می‌بریم:

```
100 CLS
200 HOME
```

برای آگاهی از این که در یک دستگاه کدام دستور به کار می‌رود، و نیز آن دستور دقیقاً چه کاری را انجام می‌دهد، باید به کتاب راهنمای دستگاه مراجعه کرد. زیرا، به عنوان نمونه، دستگاههای مختلف پس از اجرای این

دستور، مکان‌نما را در جاهای مختلفی از صفحه قرار می‌دهند.

ساختن نگاره‌هایی که در بازی‌هایی همچون «مهاجمان فضایی» به کار می‌رود، بیشتر کار هنرمندان است تا کار برنامه‌نویسان. خوانندگانی که بخواهند در این زمینه اطلاعات بیشتری به دست آورند باید به کتاب‌های خاص آن مراجعه نمایند. با این حال، در زیر چند برنامهٔ ساده را برای نشان دادن بعضی از امکانات نگاره‌سازی، به دست داده‌ایم.

رسم یک منحنی خاص، شاید ساده‌ترین نمونهٔ نگاره‌سازی باشد. برنامهٔ زیر را برای رسم تابع سینوسی مقدارهایی که در فاصلهٔ صفر و 2π قرار دارند، نوشته‌ایم. صفحهٔ نمایش، چنان که در شکل ۱— ۱۰ دیدیم، داری ۲۸۱۶ یاختهٔ تصویری می‌باشد:

```
100 FOR J = 0 TO 63
110 PLOT J, 20*SIN (J/32*3.14) + 22
120 NEXT J
130 END
```

باید دقت کرد که در دستور نگاره‌گذاری (PLOT) مقدارهای داده شده یاخته‌ای را در داخل صفحه نمایش دستگاه مورد استفاده، تعیین نمایند. گاهی لازم است که مقدارها را به نسبتی کوچک کرد تا در داخل محدودهٔ صفحه نمایش قرار گیرند. برای مثال، اگر بخواهیم نمودار میله‌ای مقدارهایی را که در فاصلهٔ صفر تا ۱۰۰ قرار دادند رسم نمائیم، باید از روش زیر استفاده نمائیم (L شمارهٔ سطر و V اندازهٔ «طول میله» را نشان می‌دهد.)

```
410 FOR J = 0 TO INT (V*32/100) - 1
420 PLOT J,L
430 NEXT J
```

عبارت سطر ۴۱۰ مقدار V را چنان مقیاس‌بندی می‌نماید که همواره در فاصلهٔ صفر تا ۳۲، یعنی در محدودهٔ ستون‌های صفحهٔ نمایش، قرار گیرد.

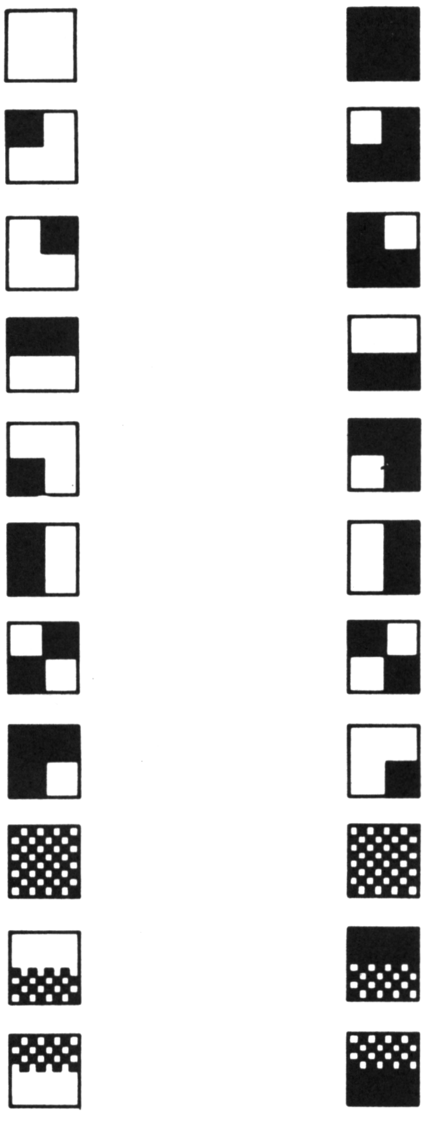
برنامه‌ای که «سرخک» نامیده می‌شود یک بار به صورت تصادفی

تعدادی نقطه را روی صفحهٔ نمایش می‌گذارد، و بار دیگر آن‌ها را پاک می‌کند. یکی از روش‌های نوشتن این برنامه چنین است:

```
100 REM FILL THE SCREEN
110 FOR J = 1 TO 2816
120 PLOT INT (RND*64),INT(RND*44)
130 NEXT J
140 REM NOW EMPTY THE SCREEN
150 FOR J = 1 TO 2816
160 UNPLOT INT (RND*64),INT(RND*44)
170 NEXT J
180 END
```

توجه نمائید که هرچند دستور نگاره‌گذاری (PLOT) ۲۸۱۶ بار تکرار می‌شود، ولی معلوم نیست که تمام یاخته‌های تصویری سیاه شوند. زیرا استفاده از تابع‌های عدد تصادفی (RND) و عدد صحیح باعث می‌شود که بعضی از یاخته‌ها دو بار سیاه شوند. همین اتفاق در دستور نگاره‌زدایی (UNPLOT) هم خواهد افتاد و در نتیجه بعضی از یاخته‌های سیاه شده همچنان برجای خواهند ماند. البته شما می‌توانید برنامه‌ای بنویسید که همه صفحه را کاملاً سیاه کرده و بعد آن را سفید نماید.

بعضی از دستگاهها یاخته‌های تصویری مربعی شکلی را به کار می‌برند که اندازهٔ آن‌ها به اندازهٔ جای نشانه‌های نوشتاری بوده و هر کدام نشانهٔ نگاره‌ای خاصی را نمایش می‌دهند. نمونهٔ این نشانه‌ها را در شکل ۲—۱۰ می‌بینید:



شکل ۲ــ ۱۰ نشانه‌های نگاره‌ای

نشانه های شطرنجی نوعی رنگ خاکستری‌نما را پدید می‌آورند، که چنانچه دستگاه نمایش سیاه و سفید باشد، تعداد رنگ‌های آن را به سه می‌رساند. برای بکار بردن این نشانه ها باید از دستور چاپ (PRINT) یا دستور بایت گذاری (POKE) استفاده کرد تا از دستور نگاره گذاری (PLOT). با ترکیب نشانه های نگاره‌ای می‌توان نمودارهای میله‌ای را به آسانی ساخت. اگر نگارهٔ موردنظر در متغیر G$ باشد و درازای میله (که کمتر از ۳۲ است) در L برنامهٔ زیر را برای رسم نمودار میله‌ای می‌توان به کار برد:

```
200 FOR J = 1 TO L
210 PRINT G$;
220 NEXT J
230 PRINT
```

دستور چاپ سطر ۲۳۰ را برای این به کار برده‌ایم که بعد از رسم نمودار، مکان‌نما را به سطر بعدی ببرد.

چنان که در شکل ۲— ۱۰ می‌بینید دو نشانهٔ کاملاً شطرنجی وجود دارد. این دو نشانه کمی با هم تفاوت دارند و برای این درست شده‌اند که اگر در کنار هم قرار گیرند محل وصل آن‌ها مشخّص باشد (زیرا اگر یکی از آن‌ها را در کنار خودش قرار دهیم چنین نیست.)

نگاره‌سازی ریزنگاشت

دستگاههای جدیدتر به جای نگاره‌سازی نه چندان دقیق (یا درشت نگاشت) که پیش از این دیدیم می‌توانند به نگاره‌سازی ریزنگاشت (High resolution) یا دقیق بپردازند. به عبارت ساده‌تر، این دستگاهها تصویرها را با جزئیات بیشتر تهیه می‌نمایند.

در صفحهٔ نمایش ریزنگاشت، بیش از ۴۰۰۰۰ یاختهٔ تصویری وجود دارد. که در بیشتر دستگاهها، رنگی نیز می‌باشند. روشن است که با این دستگاهها می‌توان تصویرها را با تفضیل بیشتر فراهم آورد. از این روش در برنامه های بازی نیز استفاده می‌شود.

دستگاههای ریزنگاشت، بیسیکی را به کار می‌برند که دستورهایی خاص آن‌ها دارد؛ همچون دستورهای «رنگ‌آمیزی ریز» (HCOLOR) و «نگاره گذاری ریز» (HPLOT)

کار این دستورها با دستورهایی که دربارهٔ «رنگ‌آمیزی» و «نگاره گذاری» معمولی دیدیم تفاوتی ندارد، جز این که باید آن‌ها را با جزئیات بیشتری به کار برد.

بخش یازدهم

یادآوری‌های برنامه‌نویسی

گاهی لازم است برنامه‌ای را که پیش از این نوشته‌ایم تغییر داده یا دستکاری نمائیم. یکی از دشوارترین چیزها، برای این کار، شاید به‌یاد آوردن همه منظورهایی باشد که از نوشتن برنامه در سر داشته‌ایم. هرچه از زمان نوشتن برنامه بیشتر دور شده باشیم این دشواری بیشتر می‌شود. سختی کار چند برابر می‌شود اگر بخواهید برنامه‌ای را که شخص دیگری نوشته است شما تغییر دهید.

به‌کار بستن چند رهنمود ساده، در هنگام برنامه‌سازی، کار شما را در هنگام تغییرات بعدی بسیار سریعتر و آسانتر خواهد کرد. اگر این گفته را باور ندارید یکی از برنامه‌هایی را که قبلاً نوشته‌اید تغییر دهید، یا برنامه‌ای را که کسی دیگر نوشته است بررسی نمائید، تا درستی آن را دریابید. هرچند برنامه‌ها درست «مستند» شده باشند باز کار دشوار است، ولی اگر درست مستندسازی نشده باشند، یا اصلاً مستند نشده باشند، در اینصورت واقعاً نمی‌شود کاری انجام داد. چگونه باید، و یا بهتر است، که برنامه‌ای را مستند کنیم؟ این کار را می‌توان به‌سه بخش جدا از هم تقسیم کرد:

- شرح برنامه
- گردش نمای برنامه
- نوشتن برنامه

شرح برنامه

در این بخش کاری را که برنامه باید انجام دهد به‌طور کلی شرح می‌دهیم ولی جزئیات آن را به نوشتن برنامه واگذار می‌کنیم. برای این کار هدف‌های کلی برنامه، خلاصهٔ کاری که برنامه انجام می‌دهد، و رابطهٔ آن را با برنامه‌های دیگر شرح می‌دهیم. چنانچه از روش‌ها و شیوه‌های غیرمعمولی به‌ناگزیر استفاده کرده‌ایم، باید آن‌ها را در شرح برنامه کاملاً روشن کنیم.

شرح برنامه باید گویا و مختصر باشد. در زیر شرح برنامه‌ای را که خود جزئی از یک سیستم «نام و آدرس مشتریان» است، برای نمونه می‌آوریم:

درست‌سازی نام‌های جدید

نام‌ها و نشانی‌های جدید خوانده و بررسی می‌شوند. اگر نام‌ها درست باشند، وارسی شده که دوگانه (تکراری) نباشند. **چنانچه یگانه** باشند در پرونده اصلی نام و نشانی نوشته شوند. نام‌ها و نشانی‌های **نادرست** و دوگانه، برای این‌که بعداً چاپ شوند، علامت‌گذاری می‌شوند. نام و نشانی‌های درست از صفحه کلید دریافت شده و در پروندهٔ میانجی نوشته می‌شود، و سپس برای به‌هنگام درآوردن پروندهٔ اصلی نام و نشانی به کار می‌روند.

شرح برنامه‌های ساده آسان است. اگر محدودیت زمانی مانع نباشد، بهتر است که به‌جای یک برنامهٔ بزرگ، چندین برنامهٔ کوچک نوشته شود. نوشتن و غلط‌گیری یک برنامهٔ کوچک بسیار آسانتر است. اگر مجبور باشیم که تنها یک برنامهٔ بزرگ بنویسیم، در این صورت باید آن را به بخش‌های منطقی کوچکتر تقسیم نمائیم، تا بتوانیم نوشتن آن را به‌درستی به‌انجام برسانیم.

پیش از این روش استفاده از زیرروال‌ها، و ارزش آن‌ها را در برنامه‌نویسی، دیده‌ایم. هنگامی که می‌خواهیم برنامهٔ بزرگی بنویسیم، پیش از هرچیز باید کتابخانهٔ «زیرروال‌ها» را بررسی کنیم تا ببینیم که آیا چیزی که به‌کار ما بیاید در آن هست یا نیست. مثلاً، در برنامه‌ای که بیش از این

شرح دادیم، بسیار احتمال دارد که پیش از این زیرروال‌هایی برای وارسی درستی نام‌ها و نشانی‌ها، و همچنین به‌هنگام در آوردن پروندهٔ اصلی نام و نشانی، نوشته شده باشد. بهتر است که در نوشتن برنامه‌های تازه هم از زیرروال استفاده شود، به‌خصوص اگر احتمال داده شود که آن‌ها را بتوان در برنامه‌های دیگر نیز مورد استفاده قرار داد. در مثال بالا، برای خواندن نام‌ها و نشانی‌ها از صفحه کلید، می‌توان یک زیرروال به کار برد.

پس از تعیین قسمت‌هایی از برنامه که می‌توان آن‌ها را به‌صورت زیرروال درآورد، کار بعدی تهیّه نمودار گروهی (Block diagram) از بخش‌های منطقی برنامه می‌باشد. در برنامه‌های کوچک این کار را می‌توان به‌صورت ذهنی انجام داد. مثلاً در مثال پیشین می‌توان بخش‌های زیر را در نظر گرفت:

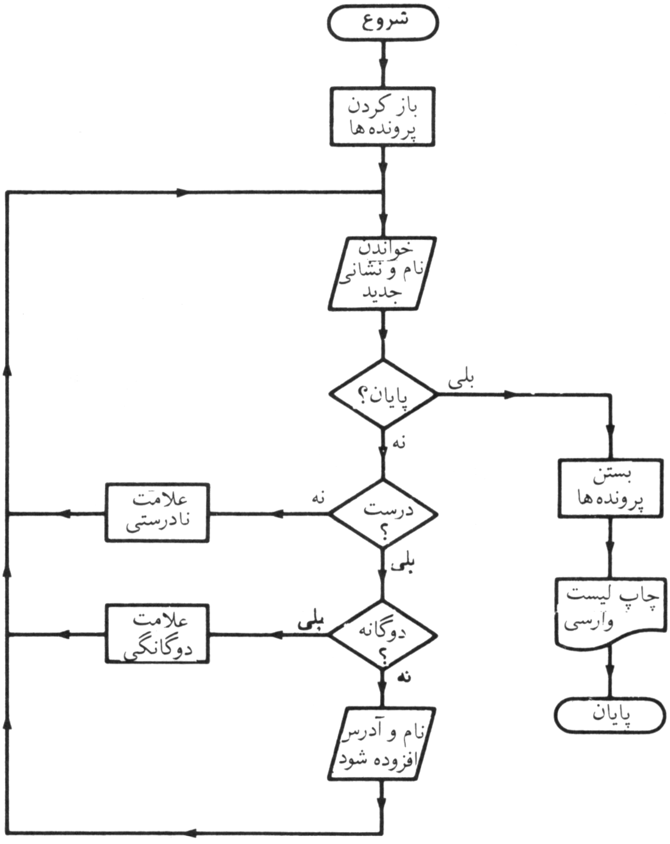
- **کارهای آغازی**
- **ورودی**
- **پردازش**
- **خروجی**

هر بخش جدای برنامه را می‌توان به‌تنهایی نوشت و آزمایش کرد تا کار برنامه‌نویسی آسانتر شود. با به کار بردن گردش‌نما *، کار برنامه‌نویسی بازهم آسانتر می‌شود.

گردش‌نمای برنامه

نمایش نموداری کارها و روند منطقی برنامه را «گردش‌نمای برنامه» می‌نامیم. بازهم، اگر برنامه بسیار کوچک باشد می‌توان گردش‌نمای آن را تنها در ذهن آماده ساخت. گردش‌نما را می‌توان به‌دو شیوهٔ فشرده، — که تنها نشان دهنده کارهای اصلی برنامه باشد— و یا گسترده، که هر مرحله یا پردازش خاص را نشان دهد، کشید. بسیار مهم است که بتوان میان فشرده‌گویی، که بیشتر مورد پسند است، و گستردگی، که آن‌هم مورد نیاز است، هماهنگی برقرار کرد. در کشیدن گردش‌نما لازم نیست که برای هر

عبارت برنامه جعبه‌ای رسم شود. گردش‌نما باید تنها نشان دهندهٔ کارهای اصلی و رابطهٔ میان آن‌ها باشد. در شکل ۱ـ۱۱ گردش‌نمای برنامهٔ به‌هنگام درآوردن پروندهٔ نام و نشانی را، که پیش از این شرح دادیم، می‌بینید.



شکل ۱ـ۱۱ گردش‌نمای برنامهٔ به‌هنگام درآوردن پروندهٔ نام و نشانی

برنامه‌نویسی

بهترین ویژگی برنامه‌سازیِ خوب آن است که برنامه خوب و خوش‌طرح نوشته شود. هر برنامه باید به‌بخش‌های منطقی، که به‌دو روش شناسانده می‌شوند، شکسته شود. نخست، به‌وسیلهٔ شمارهٔ سطرها؛ مثلاً می‌توان هربخش منطقی را با مضربی از هزار شروع کرد (اگر بین دو شماره سطر زیاد فاصله باشد اشکالی ندارد). دوم، با افزودن «یادآوری»* به‌برنامه. برای این کار از عبارت یادآوری (REM) استفاده می‌شود:

```
100 REM ARRAY FOR MONTHLY SALES
110 DIM M(13)
120 REM ARRAY FOR YEARLY TOTALS BY AREA
130 DIM A(6)
.
.
.
300 INPUT "NEXT AREA";N
310 REM AN ANSWER OF 99 TO END
320 IF N = 99 THEN 600
```

عبارت یادآوری را در هرجا که بتوان یک عبارت معمولی بیسیک را نوشت، می‌توان آورد، و نیز می‌تواند عبارت مورد مراجعهٔ دستورهای جهش (GOTO) و رفتن به زیرروال (GOSUB) باشد:

```
100 GO TO 200
.
.
.
200 REM START OF INPUT PROCESSING
.
.
.
280 GOSUB 2000
.
.
320 GOSUB 2500
.
```

```
.
.
2000 REM SUBROUTINE TO VALIDATE ACCOUNT NO
2010 REM ENTRY WITH ACC NO IN A
2020 REM FLAG F SET IF INVALID
.
.
.
2500 REM SUBROUTINE TO CLEAR SCREEN
2510 REM IF FLAG S IS SET CURSOR IS
2520 REM MOVED TO TOP OF SCREEN
.
.
```

در بعضی از دستگاهها می‌توان عبارت یادآوری را به‌سطرهایی هم که دستورهایی دیگری دارند افزود. برای این کار، باید دستور قابل اجراءِ را از عبارت یادآوری، مثلاً با به کار بردن «دو نقطه» جدا کرد.

```
470 IF F = 1 THEN 520: REM IS ERROR FLAG ON
480 C = C + 1: REM COUNT NO OF VALID RECORDS
490 GOSUB 2650: REM GET NEXT INPUT RECORD
```

در به کار بردن عبارت یادآوری نباید زیاده‌روی کرد، زیرا لازم نیست که این عبارت را به‌هر سطری افزود. با استفاده درست از عبارت یادآوری، پیگیری برنامه‌ها بسیار آسانتر می‌شود. فراموش نکنید که کاربرد متغیرها را یادآوری نمائید:

```
100 DIM S(13):REM MONTHLY SALES TOTALS
110 DIM A$(20):REM SALESMEN'S NAMES
120 DIM Q(5): REM QUARTERLY SALES TOTALS
130 REM VARIABLE USAGE –
140 REM Y IS YEARLY SALES TOTAL
150 REM F IS ERROR FLAG FROM SUBROUTINE
160 REM C IS NO OF VALID RECORDS READ
170 REM W IS WEEK'S SALES
180 REM S IS SALESMAN NUMBER
```

هنگامی که برنامه به‌چند بخش منطقی شکسته شده باشد، می‌توان برنامهٔ اصلی بسیار ساده‌ای نوشت که در آن زیرروال‌هایی که پردازش‌های پردامنه را انجام می‌دهند فراخوانده شوند. این روش سودی دوگانه به‌همراه دارد: نخست این‌که می‌توان هرکار منطقی را جداگانه برنامه‌نویسی و آزمایش کرد (چنان‌که پیش از این گفتیم)؛ و دیگر این‌که برنامهٔ اصلی را —بسته به‌گردش‌نمای گروهی آن— می‌توان به‌سادگی فهمید و به‌آسانی تغییر داد.

برنامهٔ اصلی وارسی پرونده نام و نشانی را، چنان‌که دو شکل ۱—۱۱ نشان داده شد، می‌توان به‌روش زیر نوشت:

```
100 REM DEFINES VARIABLES ETC
.
.
.
.
200 REM INITIALIZATION ROUTINE
210 F = 1
220 A = 0
230 R = 0
240 GOSUB 2000: REM OPEN FILES
250 REM MAIN LINE START
260 GOSUB 2500: REM GET NEW N & A
270 IF F = 0 THEN 500
280 GOSUB 3000: REM TEST VALIDITY
290 IF F = 1 THEN 320
300 GOSUB 3900: REM FLAG AS INVALID
310 GO TO 250
320 GOSUB 5000: REM TEST FOR DUPLICATION
330 IF F = 1 THEN 360
340 GOSUB 3700: REM FLAG AS DUPE
350 GO TO 250
360 GOSUB 6000: REM ADD TO N & A FILE
370 GO TO 250
500 F = 2
510 GOSUB 2000: REM CLOSE FILES
520 GOSUB 7500: REM PRINT CHECK TOTALS
530 GO TO 999: REM END OF MAINLINE
```

اکنون می‌توان به‌آسانی این برنامه را با نمودار شکل ۱ــ ۱۱ پیگیری کرد.

صرفه‌جویی در حافظه

بیشتر ریز کامپیوترها دارای حافظه‌ای کوچک می‌باشند. این مشکل به‌خصوص دامنگیر کامپیوترهای خانگی است که باید ارزانتر به‌فروش برسند. اگر نتوانیم به‌علت کمبود حافظه همهٔ دستورهای برنامه را یک‌جا به‌کامپیوتر بدهیم، آن‌گاه اندازهٔ کاری که برنامه می‌تواند انجام دهد بسیار محدودتر خواهد شد. برای کاستن از این مشکل، و صرفه‌جویی در حافظه، روش‌هایی وجود دارد که باید آن‌ها را به‌کار بست. زیاده‌روی در مصرف حافظه از ساختمان خاص عبارت‌های بیسیک ریشه می‌گیرد. اگر بتوان این ساختمان را از میان برد، یا آن را کوچکتر کرد، آن‌گاه این زیاده‌روی از میان می‌رود، یا کمتر می‌شود. هرچند باید یادآوری کرد که صرفه‌جویی بسیار در حافظه باعث پیچیدگی برنامه خواهد شد، از این‌رو تنها هنگامی که لازم است باید به‌این کار دست زد.

زبان بیسیک برای هر شمارهٔ سطر چندین جای حافظه را می‌گیرد. از این‌رو، اگر در یک سطر چند عبارت نوشته شود، این خود باعث صرفه‌جویی در حافظه خواهد شد:

```
100 A=0:B=1:C=2
```

چنان‌که در مورد عبارت یادآوری دیدیم، این‌جا هم برای نوشتن چند عبارت در یک سطر، آن‌ها را با دو نقطه از یکدیگر جدا می‌کنیم. تعداد عبارت‌هایی که بعد از یک شمارهٔ سطر می‌آیند، به‌طول سطرهای صفحهٔ نمایش بستگی ندارند. در بیشتر گونه‌های زبان بیسیک، طول سطر برنامه می‌تواند بزرگتر از طول سطر صفحهٔ نمایش باشد (گاهی تا بیش از ۲۰۰ نشانه را دربر می‌گیرد). همهٔ عبارت‌های زیر در سطر شمارهٔ ٦٤٠ برنامه نوشته شده‌اند:

```
640 A = B*2 + 1:IF A = 15 THEN 650:
    R = 0: PRINT "OUT OF RANGE":
    GO TO 720
```

ولی این روش «چند عبارتی» کار ویرایش و تغییرات بعدی برنامه را مقداری مشکلتر خواهد کرد، زیرا گاهی برای افزایش چند نشانه مجبور می‌شویم که همهٔ سطر را دوباره از اوّل ماشین کنیم.

هرچند پیش از این گفتیم که افزودن عبارت‌های یادآوری کار خواندن برنامه را آسانتر می‌کند، ولی گاهی ناچاریم که از این کار خودداری کنیم. عبارت یادآوری برای هر نشانه‌ای که به‌کار می‌برد جایی از حافظه را می‌گیرد. از این‌رو اگر ناچار شدیم یا باید یادآوری‌ها را بسیار کوتاه کنیم، یا برای افزودن بر حافظهٔ مورد استفاده از آن‌ها چشم‌پوشی نمائیم.

ثابت‌ها نیز از حافظه مصرف می‌کنند و از این‌رو باید در به‌کار بردن آن‌ها (به‌خصوص هنگامی که با کمی جا روبه‌رو هستیم، زیاده‌روی نکنیم. اگر ثابت خاصی را بارها در برنامه به‌کار ببریم، بهتر است نخست آن را به‌یک متغیّر واگذار کنیم، سپس هرجا لازم است از آن متغیّر استفاده نمائیم. برای مثال، در یک برنامهٔ مثلثاتی می‌توان مقدار π را در یک متغیّر گذاشت و سپس هرجا که لازم است آن را به‌کار برد:

```
100 P = 3.1416
.
.
.
200 R = P*D
.
.
.
300 A = P*R↑2
```

متغیرهای اعشاری، همچون π در مثال بالا، معمولاً از متغیرهای صحیح جای بیشتری می‌گیرند. از این‌رو عدد ۲۷ بر عدد ۲۷/۰، از این جهت، برتری دارد.

هر متغیری که در برنامه به‌کار رود، حال چه آشکارا معرفی شده باشد (مثلاً در عبارت بعد گذاری (DIM)، و چه سربسته (مثلاً در عبارت

واگذاری (LET)، به‌جایی در حافظه نیاز دارد. از این‌رو، اگر متغیّری را در بخش‌های گوناگون برنامه مورد استفادهٔ دوباره قرار دهیم، به‌جای این که متغیر تازه‌ای معرفی کنیم، این کار باعث صرفه‌جویی در حافظه خواهد شد. نمونه‌ای از این استفادهٔ دوباره را در برنامهٔ اصلی به‌هنگام درآوردن پروندهٔ نام و نشانی، که در آن متغیر F را در همهٔ زیرروال‌ها به‌کار برده‌ایم، می‌توان دید. اگر آرایه‌ای را چنان که پیش از این دیدیم مورد استفاده قرار دهیم، خانه شمارهٔ صفر آن به‌کار گرفته نمی‌شود. این خانه را می‌توان، همچون متغیّری جداگانه، به‌کار گرفت (تا از جایی که برای آن پیش‌بینی شده است استفاده کرده باشیم). در مثال زیر، خانه شمارهٔ صفر آرایه M را (که به‌صورت M(13) معرفی شده است) به‌عنوان «نشان نادرستی» به‌کار برده‌ایم:

```
.
.
260 GOSUB 2000
270 IF M(0) = 1 THEN 400
.
.
.
310 GOSUB 3600
320 IF M(0) = 1 THEN 460
```

در بعضی از دستگاهها، بعد از این که متغیّری مورد استفاده قرار گرفت، همچنان در حافظه برجا می‌ماند. این روش، به‌خصوص در مورد ثابت‌ها و متغیرهای حرفی، مشکلاتی به‌همراه می‌آورد. برای آزاد ساختن حافظه از این متغیّرهای بی‌کار، باید فرمان مناسبی به‌دستگاه داده شود، که معمولاً چنین است.

```
100 CLEAR
```

دستگاههای مختلف، شکل‌های گوناگونی از این عبارت را برای بازنشانی متغیرها و آزاد ساختن حافظه از متغیرهای بی‌کار، به‌کار می‌برند.

ویرایش

هنگامی که برنامه را از صفحه کلید به دستگاه می‌دهیم، ممکن است در ماشین کردن دچار اشتباه شویم. علاوه‌بر این اشتباه ساده، احتمال دارد برنامه دارای اشتباههای منطقی باشد که پیدا کردن آن‌ها چندان آسان نیست.

پیش از این گفتیم که می‌توانیم یک سطر اشتباه را، با دادن شمارهٔ آن و ماشین کردن عبارت درست، اصلاح کنیم. برای پاک کردن یک سطر از برنامه، کافی است تنها شمارهٔ آن را ماشین کنیم. به‌عبارت‌های زیر توجه کنید:

```
100 DIM A(10)
100 DIM A(11)
110 REM INITIALISE
120 A = 0
130 B = 0
105 DIM Y(6)
140 INPUT A$
150 PRINT "START OF PROCESSING"
140
```

همچنان که این برنامه را به‌ماشین می‌دهیم، می‌توان با تکرار شمارهٔ سطرها، اشتباههای برنامه را اصلاح کرد. حال اگر لیست برنامه را بگیریم، آن رابه‌شکل زیر خواهیم یافت (می‌دانید که فرمان لیست‌گیری (LIST) از همهٔ برنامه لیست می‌گیرد، اگراین فرمان به‌صورت LIST150 باشد تنها از سطر ۱۵۰ لیست گرفته می‌شود):

```
LIST
100 DIM A (11)
105 DIM Y(6)
110 REM INITIALISE
120 A = 0
130 B = 0
150 PRINT "START OF PROCESSING"
```

برای ویرایش برنامه و داده‌ها، راه دیگری نیز وجود دارد. می‌توان

لیست برنامه را در روی صفحه چاپ کرد. سپس مکان‌نما را با استفاده از دکمه‌هایی که آن را در چهار طرف راست، چپ، بالا و پائین حرکت می‌دهند، در زیر نشانهٔ نادرست قرار دارد (این دکمه‌ها با علامت پیکانی که در روی آن‌ها است، جهت حرکت مکان‌نما را نشان می‌دهند). حال می‌توان نشانهٔ درست را ماشین کرد تا برنامه اصلاح شود.

در هنگام ماشین کردن عبارت‌ها، چنانچه حرفی را اشتباه ماشین کردیم با دکمهٔ «پاک کن» (Rubout) یا «چپ‌رو» (که گاهی به‌آن «پس‌رو» نیز گفته می‌شود) می‌توان آن را پاک کرد. چندین نشانهٔ پیوسته به‌هم را می‌توان با چندبار فشار دادن دکمهٔ «پاک کن» پاک کرد. مکان‌نما برای نشان دادن نتیجهٔ کار همزمان به‌عقب برمی‌گردد.

بخش دوازدهم
چند برنامهٔ بازی ساده

در این بخش چند برنامهٔ بازی ساده را، که به‌زبان بیسیک نوشته شده‌اند، شرح می‌دهیم. دراین برنامه‌ها تنها دستورهایی به‌کار رفته‌اند که در بیشتر ریزکامپیوترها وجود دارند. از این‌رو هریک از این برنامه‌ها را می‌توان به‌آسانی، در هر دستگاهی، به‌کار بست. در هر برنامه چندین‌بار از عبارت یادآوری (REM) برای آسانتر کردن پیگیری آن، استفاده شده است؛ هرچند دارندگان دستگاههای کوچک به‌یاد خواهند داشت که این عبارت‌ها جایی را در حافظه کامپیوتر می‌گیرند، از این‌رو شاید ناچار شوند که از آن‌ها چشم‌پوشی نمایند (در بعضی از دستگاهها باید دستورهای واگذاری را با واژهٔ LET به‌کار برد، مثلاً: LET V = − 5).

بازی شماره ۱

این برنامه نگارشی از برنامهٔ قدیمی «ماه‌نشین» است، که احتمالاً در ده سال گذشته در همهٔ کتاب‌های برنامه‌نویسی به‌کار رفته است. هدف آن است که ماه‌نشین را با سرعتی میان صفر تا ۱۰۱- در ماه فرود آوریم.

هرچه سرعت بیشتر باشد آسیب‌های بزرگتری برآن وارد خواهد شد. برای کاستن از نیروی گرانشی (که در آغاز برنامه برابر ۵۰- است)، موشک‌های کاهنده به‌مدت ۱ تا ۱۰ ثانیه و با فشاری میان صفر تا ۹۹ روشن می‌شوند.

```
110 REM MOON LANDER
120 REM V IS DESCENT VELOCITY
130 REM H IS HEIGHT ABOVE SURFACE
140 REM R IS FUEL REMAINING
150 REM SET INITIAL VALUES
160 V = -50
170 H = 1000
180 R = 5000
190 GO TO 340
200 REM DETERMINE PILOT INSTRUCTIONS
210 PRINT "BURN LEVEL (0-99)";
220 INPUT F
230 PRINT
240 PRINT "LENGTH OF BURN (1-10)";
250 INPUT T
260 REM CLEAR DISPLAY
270 CLS
280 IF F *T < R/10 THEN 310
290 F = R/10*T
300 R = R - F*T*10
310 B = F - 32
320 H = B*T↑2 + V*T + H
330 V = 2*B*T + V
340 PRINT "SPEED","HEIGHT","FUEL"
350 PRINT V,H,R
360 PRINT
370 IF H > 0 THEN 210
380 IF V < -250 THEN 530
390 IF V < -200 THEN 510
400 IF V < -150 THEN 490
410 IF V < -100 THEN 470
420 IF V < -50 THEN 450
430 PRINT "SMOOTH LANDING, WELL DONE"
440 GO TO 540
450 PRINT "BUMPY, BUT SAFE"
460 GO TO 540
470 PRINT "SOME DAMAGE—YOU MIGHT SURVIVE"
480 GO TO 540
490 PRINT "YOU WON'T BE GOING HOME"
500 GO TO 540
510 PRINT "WHAT A BIG CRATER!"
```

```
520 GO TO 540
530 PRINT "B A N G"
540 PRINT
550 PRINT "TRY AGAIN"
560 INPUT A$
570 IF A$ = "YES" THEN 150
990 END
```

بازی شماره ۲

این برنامه ویژهٔ ستاره‌شناسان است (!) زیرا از روی تاریخ تولد افراد برجی را که به‌دنیا آمده‌اند محاسبه می‌کند. اگر برج‌ها را فراموش کرده‌اید، این‌ها هستند:

۱— دی (بزغاله یا جدی	Capricorn)	۲۳ دسامبر تا ۱۹ ژانویه
۲— بهمن (دلو	Aquarius)	۲۰ ژانویه — ۱۹ فوریه
۳— اسفند (ماهی یا حوت	Pisces)	۲۰ فوریه — ۲۰ مارس
٤— فروردین (بره یا حمل	Aries)	۲۱ مارس — ۱۹ آوریل
۵— اردیبهشت (گاو یا ثور	Taurus)	۲۰ آوریل — ۲۰ مه
٦— خرداد (دوپیکر یا جوزا	Gemini)	۲۱ مه — ۲۰ ژوئن
۷— تیر (خرچنگ یا سرطان	Canser)	۲۱ ژوئن — ۲۱ ژوئیه
۸— مرداد (شیر یا اسد	Leo)	۲۲ ژوئیه — ۲۲ اوت
۹— شهریور (خوشه یا سنبله	Virgo)	۲۳ اوت -- ۲۲ سپتامبر
۱۰— مهر (ترازو یا میزان	Libra)	۲۳ سپتامبر — ۲۲ اکتبر
۱۱— آبان (کژدم یا عقرب	Scorpio)	۲۳ اکتبر — ۲۱ نوامبر
۱۲— آذر (کمان یا قوس	Sagittarius)	۲۲ نوامبر — ۲۲ دسامبر

(توجه داشته باشید که ماههای ایرانی با برج‌ها، گاهی یکی دو روز اختلاف دارند. م)

```
100 REM ZODIAC SIGN PROGRAM
110 DIM Z$ (13)
120 REM SET UP NAMES OF SIGNS
130 REM IF YOUR SYSTEM DOES NOT
140 REM HAVE READ/DATA, USE MULTIPLE
```

```
150 REM LET STATEMENTS
160 FOR J = 1 TO 12
170 READ Z$ (J)
180 NEXT J
190 REM CLEAR DISPLAY
200 CLS
210 PRINT "WHAT IS YOUR BIRTHDAY (DAY,
      MONTH)";
220 INPUT D,M
230 R = M
240 L = 20
250 REM IF YOU DO NOT HAVE ON GO TO,
      STATEMENT,
260 REM USE IF STATEMENTS
270 ON M GO TO 310,310,300,310,300,300,290,280,280,
      280,290,280
280 L = L + 1
290 L = L + 1
300 L = L + 1
310 IF D < L THEN 330
320 R = R + 1
330 IF R < = 12 THEN 350
340 R = 1
350 PRINT "YOUR ZODIACAL SIGN IS ";Z$(R)
360 GO TO 200
370 DATA "CAPRICORN","AQUARIUS","PISCES",
      "ARIES"
380 DATA "TAURUS","GEMINI","CANCER","LEO"
390 DATA "VIRGO","LIBRA","SCORPIO",
      "SAGITTARIUS"
999 END
```

برنامهٔ شماره ۳

در این بازی کامپیوتر عددی را انتخاب می‌کند و شما باید حدس بزنید که آن عدد چیست. عددی را که باید حدس بزنید، عددی است تصادفی میان صفر تاعددی که شما تعیین کرده‌اید. بعد از هر حدس زدن، کامپیوتر به‌شما خواهد گفت که عدد حدسی شما از عدد انتخابی بزرگتر است یا کوچکتر. وقتی که حدس شما درست بود، عدد چاپ می‌شود.

```
100 REM GUESS A NUMBER GAME
110 PRINT "WHAT IS THE HIGHEST NUMBER TO
    USE";
120 INPUT H
130 N = INT (H*RND(X))
135 C = 0
140 PRINT
150 PRINT "GUESS THE NUMBER I HAVE PICKED"
160 PRINT "BETWEEN 0 AND ";H
170 PRINT
180 PRINT "YOUR GUESS"
190 INPUT G
200 C = C + 1
210 IF G = N THEN 270
220 IF G < N THEN 250
230 PRINT "TOO HIGH"
240 GO TO 170
250 PRINT "TOO LOW"
260 GO TO 170
270 PRINT "CORRECT IN ";C;" TRIES"
280 PRINT "TRY AGAIN";
290 INPUT A$
300 IF A$ = "YES" THEN 110
999 END
```

بازی شماره ۴

این برنامهٔ «تاس بازی» بازی دلخواه کسانی است که حوصلهٔ بازی‌های طولانی را ندارند. برنامه از شما می‌پرسد که چند دور می خواهید بازی کند، سپس آن تعداد بازی را انجام می‌دهد، بی‌آنکه نتیجه هردوربازی‌را در پایان آن به‌شما بگوید. پس از این که بازی به‌پایان رسید، آن گاه شما را از تعداد بازی‌ها، و نیز تعدادی که شما برده‌اید، آگاه می‌سازد! (روش این بازی در عمل و به‌طور خلاصه چنین است: بازی‌کن تاس‌ها را می‌اندازد، اگر مجموع دوتاس ۷ یا ۱۱ باشد، او برده است. اگر ۲، ۳ یا ۱۲ باشد باخته است. اگر عددی جز این‌ها باشد، نوعی امتیاز است. بازی‌کن باید دوباره تاس‌ها را بیاندازد، و این کار را آنقدر تکرار کند که یا ۷ بیاید تا ببازد، یا دوباره همان مقدار قبلی را بیاورد تا برنده شود).

```
100 REM AUTOMATIC CRAPS PLAYER
110 RANDOMISE
120 CLS
130 PRINT "HOW MANY GAMES TO PLAY";
140 INPUT N
150 W = 0
160 FOR J = 1 TO N
170 F = INT (6*RND(X)) + INT(6*RND(X)) + 2
180 IF F = 7 THEN 280
190 IF F = 11 THEN 280
200 IF F = 2 THEN 290
210 IF F = 3 THEN 290
220 IF F = 12 THEN 290
230 REM F IS NOW THE POINT
240 REM DICE ROLLED AGAIN
250 P = INT (6*RND(X)) + INT(6*RND(X)) + 2
260 IF P = 7 THEN 290
270 IF P <> F THEN 250
280 W = W + 1
290 NEXT J
300 PRINT
310 PRINT N;" GAMES PLAYED, ";W;" WINS"
320 PRINT
330 PRINT "MORE"
340 INPUT A$
350 IF A$ = "YES" THEN 110
999 END
```

بازی شمارهٔ ۵

سال‌ها است که کودکان با تعدادی چوب کبریت یا مهره به‌بازی «نیم» می‌پردازند. برای این بازی ۲۱ عدد چوب کبریت را روی میز در کنار هم می‌گذارند. دو نفر با یکدیگر بازی می‌کنند، و هریک هربار از یک تا چهار چوب کبریت را می‌تواند بردارد. هرکس آخرین چوب را بردارد بازنده است. در این بازی نفر دوم می‌تواند شیوه‌ای را به‌کار ببرد که همیشه برنده باشد. در این مثال کامپیوتر به‌جای نفر دوم بازی می‌کند، از این‌رو همیشه برنده است. این برنامه وسیلهٔ خوبی است برای کلک‌زدن به‌دوستانی که نمی‌دانند، اگر نخستین بازی‌کن باشند، هیچ‌گاه برنده نخواهد شد!

```
100 REM GAME OF NIM
110 CLS
120 PRINT" THE GAME OF NIM"
130 PRINT
140 N = 21
150 PRINT "HOW MANY DO YOU WISH TO TAKE"
160 INPUT R
170 IF R <> INT(R) THEN 300
180 IF R < 1 THEN 320
190 IF R > 4 THEN 320
200 N = N - 5
210 PRINT "I TAKE ";5 - R; " LEAVING ";N
220 IF N > 1 THEN 150
230 PRINT "YOU PICK THE LAST ONE"
240 PRINT "SO I WIN"
250 PRINT
260 PRINT "AGAIN"
270 INPUT A$
280 IF A$ = "YES" THEN 110
290 GO TO 999
300 PRINT "WHOLE NUMBERS ONLY"
310 GO TO 150
320 PRINT "NO CHEATING - 1,2,3, OR 4 ONLY!"
330 GO TO 150
999 END
```

بازی شمارهٔ ۶

این برنامه در اصلی «روالی» است که تعیین می‌کند فلان تاریخ (برحسب روز و ماه و سال) چندشنبه بوده است. در این‌جا از آن برای محاسبهٔ روزی از هفته که افراد به‌دنیا آمده‌اند استفاده کرده‌ایم، ولی آن را برای هر تاریخی می‌توان به‌کار برد. این برنامه را می‌توان به‌برنامهٔ شماره ۲ (محاسبهٔ برج تولد افراد) افزود و برنامهٔ کاملتری به‌دست آورد:

```
100 REM DISCOVER DAY OF BIRTH
110 DIM D$(8)
120 REM SET UP DAYS,USE LET IF NO READ
130 FOR J = 1 TO 7
140 READ D$(J)
150 NEXT J
```

```
160  DATA  "MONDAY","TUESDAY","WEDNESDAY",
     "THURSDAY"
170 DATA "FRIDAY", "SATURDAY", "SUNDAY"
180 CLS
190 PRINT "WHAT IS BIRTHDATE (AFTER 1752)"
200 PRINT "GIVE AS DD,MM,YYYY"
210 INPUT D,M,Y
220 IF D = 0 THEN 999
230 IF Y < 1752 THEN 190
240 GOSUB 280
250 PRINT
260 PRINT "DAY WAS ";D$(R)
270 GO TO 190
280 N = INT(0.6 + 1/M)
290 L = Y - N
300 P = M + 12*N
310 C = L/100
320 Y = INT(C)
330 Z = INT (C/4)
340 X = INT (5*L/4)
350 W = INT (13*(P + 1)/5)
360 R = W + X - Y + Z + D + 5
370 R = R -(7*INT(R/7)) + 1
380 RETURN
999 END
```

یادآوری: زیرروالی که در سطرهای ۲۸۰ تا ۳۸۰ به کار رفته است شمارهٔ روز را در هفته محاسبه کرده و در متغیر R برمی‌گرداند. تاریخ باید بعد از ۱۷۵۲ باشد و به‌صورت D , M , Y به‌ماشین داده شود.

بخش سیزدهم
برنامه‌های نمونه برای کاربردهای بازرگانی

کسانی که ریزکامپیوترها را برای کارهای بازرگانی به‌کار می‌برند، و نیز کسانی که خود را با کامپیوتر سرگرم می‌کنند. ولی جز بازی به‌کارهای دیگر کامپیوتر هم علاقه‌مند هستند، این بخش را سودمند خواهند یافت. نمی‌توان برنامه‌های کاملی را که بتوان در کاربردهای بازرگانی به‌کار بست برای نمونه آورد؛ زیرا چنین برنامه‌هایی هم بسیار بزرگ هستند، و هم بسته به‌کاری که انجام می‌دهند دارای ریزه‌کاری‌هایی ویژه آن می‌باشند. (البته برنامه‌های بازرگانی همه کاره‌ای هم نوشته شده است، ولی این برنامه‌ها نیازمند پارامترهای فراوانی هستند که باید پیش از استفاده، آن‌ها را به‌برنامه داد.)

در این بخش به‌جای آن‌که به‌شرح راه‌حل برنامه‌های خاصی بپردازیم، بیشتر شیوه‌هایی را برای برنامه‌نویسی بازرگانی پیشنهاد خواهیم کرد. از این پیشنهادها می‌توان برای نوشتن «زیرروال‌ها»، و یا تنها برای رهنمود گرفتن، استفاده کرد.

نمونه ۱ — آرایش

پیش از این، در بخش هفتم، با آرایش موجی (Ripple sort) آشنا شدیم. ولی این روش بسیار کند است، و هنگام کار با آرایه‌های بزرگ دردسر می‌آفریند. در زیرروال زیر از روش **آرایش**

لایه‌ای (Shell sort) برای آراستن فرازی آرایهٔ S استفاده شده است. متغیر N نشان دهندهٔ تعداد خانه های آرایهٔ S می‌باشد.

```
2000 REM SHELL SORT SUBROUTINE
2010 REM SORTS ARRAYS
2020 REM SIZE OF ARRAY IN N
2030 D = 1
2040 D = D*2
2050 IF D< = N THEN 2040
2060 D = INT((D-1)/2)
2070 IF D = 0 THEN 2200
2080 FOR K = 1 TO N - D
2090 J = K
2100 L = J + D
2110 REM REVERSE COMPARISON FOR DESCENDING
        SORT
2120 IF S(J) < = S(L) THEN 2180
2130 T = S(J)
2140 S(J) = S(L)
2150 S(L) = T
2160 J = J - D
2170 IF J > 0 THEN 2100
2180 NEXT K
2190 GO TO 2060
2200 RETURN
```

در سطرهای ۲۰۳۰ تا ۲۰۶۰ مقدار D را از فرمول 2^R-1 به دست آورده‌ایم: $2^R < N < 2^{R+1}$. در سطرهای ۲۱۳۰ تا ۲۱۵۰ با استفاده از متغیر T همچون انباره‌ای موقتی، مقدارهای دو خانهٔ آرایه را با یکدیگر جا به جا کرده‌ایم.

نمونهٔ ۲— آمیزش آرایه ها

برای آمیختن (Merge) دو آرایهٔ آرایش یافته (Sorted) می‌توان از یکی از شیوه های آرایش لایه ای یا موجی، که پیش از این گفتیم، استفاده کرد. زیرروال زیر دو آرایهٔ A و B را با یکدیگر می‌آمیزد. متغیرهای M و N تعداد خانه های این دو آرایه را نشان می‌دهند:

```
2500 REM MERGE SUBROUTINE
2510 REM COLLATES SORTED ARRAYS A AND B
2520 I = 1
2530 J = 1
2540 K = 1
2550 IF A (I) > = B(J) THEN 2620
2560 REM TARGET ARRAY IS C
2570 C(K) = A(I)
2580 I = I + 1
2590 IF I > M THEN 2710
2600 K = K + 1
2610 GO TO 2550
2620 C(K) = B(J)
2630 J = J + 1
2640 IF J < = N THEN 2600
2650 REM COPY REMAINDER OF ARRAY A
2660 K = K + 1
2670 C(K) = A(I)
2680 I = I + 1
2690 IF M > I THEN 2770
2700 GO TO 2650
2710 REM COPY REMAINDER OF ARRAY B
2720 K = K + 1
2730 C(K) = B(J)
2740 J = J + 1
2750 IF N > J THEN 2770
2760 GO TO 2710
2770 RETURN
```

شیوهٔ آمیزش آرایه‌ها بسیار آسان است و می‌توان کار را با مداد و کاغذ پیگیری کرد تا آن را به درستی دریافت. برنامهٔ زیر روش استفاده از زیرروال بالا را نشان می‌دهد.

```
 90 DIM A(10),B(10),C(20)
100 READ M
110 FOR L = 1 TO M
120 READ A(L)
130 NEXT L
140 READ N
150 FOR L = 1 TO N
```

```
160 READ B(L)
170 NEXT L
180 GOSUB 2500
190 FOR L = 1 TO M + N
200 PRINT C(L)
9999 END
```

نمونهٔ ۳— عددهای اول

زیرروال زیر برای تعیین عددهای اول به کار می‌رود. تعداد عددهای اولی که باید تعیین شود به‌وسیلهٔ متغیّر N به کامپیوتر اطلاع داده می‌شود. تعداد خانه‌های آرایه P نباید از N کمتر باشد.

```
 REM PRIME NUMBER SUBROUTINE
 P(1) = 1
 P(2) = 2
 P(3) = 3
 R = 3
 FOR J = 1 TO N - 3
 R = R + 2
 K = 3
 Q = INT(R/K)
 T = R - Q*K
3100 IF T = 0 THEN 3060
3110 IF K> = Q THEN 3140
3120 K = K + 2
3130 GO TO 3080
3140 P(J+3) = R
3150 NEXT J
3160 RETURN
```

اگر بخواهیم عددهای اوّل را به‌جای این که در آرایه P نگهداری کنیم، چاپ کرده یا نمایش دهیم، باید سطر ۱۳۴۰ را به‌شکل زیر تغییر دهیم:

```
3140  PRINT   R
```

برنامهٔ زیر روش فراخواندن زیرروال بالا را نشان می‌دهد:

```
100 REM DISPLAY N PRIMES
110 DIM P(100)
120 PRINT "HOW MANY PRIMES"
130 INPUT N
140 IF N > 99 THEN 120
150 PRINT
160 PRINT "THE FIRST ";N;" PRIMES ARE "
170 GOSUB 3000
180 PRINT
9999 END
```

اگر بخواهید عددهای اول چاپ شوند، سطرهای ۳۰۱۰ تا ۳۰۳۰ را نیز باید به‌شکل زیر تغییر داد:

```
3010 PRINT "1"
3020 PRINT "2"
3030 PRINT "3"
```

نمونهٔ ٤— عامل های اوّل

برنامه زیر، که باز با عددهای اوّل سروکار دارد، مقسومٌ‌علیه های اوّل هر عدد را نشان می‌دهد (یعنی نشاندهندهٔ عامل های اوّل آن عدد می‌باشد).

```
100 REM PRIME FACTOR PROGRAM
110 PRINT
120 PRINT "NUMBER TO FACTOR"
130 INPUT N
140 IF N < 1 THEN 120
150 IF N <> INT(N) THEN 120
160 PRINT
170 T = N
180 K = 1
190 K = K + 1
200 J = 0
210 Q = INT(N/K)
220 R = N - Q*K
230 IF R <> 0 THEN 270
240 N = Q
250 J = J + 1
```

```
260 GO TO 210
270 IF J = 0 THEN 300
280 PRINT " IS DIVISIBLE BY ";K;",";J;" TIMES"
290 GO TO 190
300 IF Q > K THEN 190
310 IF N = 1 THEN 110
320 IF N <> T THEN 350
330 PRINT "NUMBER IS PRIME"
340 GO TO 360
350 PRINT " IS DIVISIBLE BY ";N;",";"ONCE"
360 PRINT
370 GO TO 110
```

نمونهٔ ۵ــ زمان گذشته

در دنبالهٔ کارهایی که با تاریخ انجام دادیم، این برنامه فاصلهٔ دو تاریخ را برحسب روز محاسبه می‌کند. برای کارهایی همچون محاسبهٔ بهره این برنامه می‌تواند سودمند باشد.

```
100 REM CALCULATE ELAPSED DAYS
110 PRINT "FIRST DATE(DD,MM,YYYY)'
120 INPUT D,M,Y
130 GOSUB 1000
140 T = C
150 PRINT
160 PRINT "SECOND DATE (DD,MM,YYYY)"
170 INPUT D,M,Y
180 GOSUB 1000
190 PRINT
200 PRINT "NO OF DAYS ELAPSED IS ";ABS(T - C)
210 PRINT "MORE"
220 INPUT A$
230 IF A$ = "YES" THEN 110
240 GO TO 9999
1000 REM CALCULATE DAY NUMBER
1010 N = 0
1020 Q = Y - 100*INT(Y/100)
1030 IF M > 2 THEN 1110
1040 N = 2
1050 IF Q = 0 THEN 1090
```

```
1060 R = Q - 4*INT(Q/4)
1070 IF R <> 0 THEN 1110
1080 GO TO 1100
1090 IF Q - 4*INT(Q/4) <> 0 THEN 1110
1100 N = 1
1110 C = INT(365.25*Q) + INT(30.56*M) + N + D
1120 RETURN
9999 END
```

نمونهٔ ۶ــ تبدیل عددها

این برنامه عددهای دهدهی (پایه ۱۰) را به‌عددهای دارای پایهٔ بزرگتر (معمولاً ۱۶ برای عددهای شانزده شانزدهی) تبدیل می‌کند. از همهٔ برنامه می‌توان برای نشان دادن عددهای تبدیل شده استفاده کرد؛ همچنین زیرروالی را که از سطر ۶۰۰۰ آغاز می‌شود می‌توان در برنامه های دیگر به کار برد.

```
100 REM CONVERT NUMBER TO NEW BASE
110 DIM A (15)
120 B$ = "0123456789ABCDEFGHIJKLMN"
130 PRINT
140 PRINT "NEW BASE"
150 INPUT B
160 PRINT "FIRST & LAST NUMBERS TO CONVERT"
170 INPUT F,L
180 REM TEST FOR F < L HERE IF DESIRED
190 FOR K = F TO L
200 PRINT
210 GOSUB 6000
220 REM PRINT SUCCESSIVE NUMBERS
230 PRINT K;TAB(6);
240 FOR C = J TO 1 STEP - 1
250 PRINT MID$ (B$,A(C) + 1,1);
260 NEXT C
270 NEXT K
280 GO TO 9999
6000 REM BASE CHANGE
6010 T = K
6020 J = 1
```

```
6030 Q = INT(T/B)
6040 R = T - Q*B
6050 T = Q
6060 A(J) = R
6070 J = J + 1
6080 IF Q > = B THEN 6030
6090 A(J) = Q
6100 RETURN
9999 END
```

نمونهٔ ۷ــ آمارشناسی

بسیاری از رشته‌های بازرگانی و دانش به‌محاسبه‌های آماری نیازمندی می‌باشند. برنامهٔ زیر نشان می‌دهد که روش محاسبهٔ میانگین، انحراف استاندارد و پراش (واریانس) دسته‌ای از عددها تا چه‌حد آسان است. عددها را در آرایهٔ A ، که تعداد خانه‌های آن در N جای دارد، نگهداری کرده‌ایم.

```
100 REM STATISTICAL CALCULATIONS
110 DIM A(100)
120 REM ARRAY A HOLDS SET OF FIGURES
130 M = 0
150 FOR J = 1 TO N
160 M = M + A(J)
170 K = K + A*A
180 NEXT J
190 M = M/N
200 V = (K - N*M*M)/(N - 1)
210 D = SQR(V)
220 PRINT
230 PRINT "NUMBER OF SAMPLES    = ";N
240 PRINT "                 MEAN = ";M
250 PRINT "             VARIANCE = ";V
260 PRINT "STANDARD DEVIATION = ";D
270 PRINT
280 REM RETURN HERE IF SUBROUTINE
999 END
```

نمونهٔ ۸ــ جداسازی رقم‌ها

روال زیر برای جداسازی رقم‌های یک عدد ۶ رقمی به کار می‌رود. (البته می‌توان این تعداد را با تغییر عبارت تکرار (FOR) در سطر ۱۳۰، افزایش داد.)

```
100 INPUT "GIVE NUMBERS, 6 DIGITS OR LESS";N
110 REM TEST FOR END
120 IF N = 0 THEN 999
130 FOR J = 5 TO 0 STEP - 1
140 K = INT(N/10 ↑ J)
150 PRINT K
160 N = N - K + 10 ↑ J
170 NEXT J
180 PRINT
190 GO TO 100
999 END
```

در این نمونه رقم‌ها را یکی‌یکی چاپ کرده‌ایم. با تغییر سطر ۱۵۰ به شکل زیر، می‌توان آن‌ها را در آرایه‌ای نگهداری کرد:

```
150 A(J) = K
```

نمونهٔ ۹ــ عددهای رومی

از این برنامه که عددهای رومی را به‌عددهای معمولی (پایه ۱۰) تبدیل می‌کند می‌توان برای سرگرم کردن دوستان استفاده کرد. با این برنامه می‌توان XXVIII را به ۲۸ و MCCLXV ۱۲۶۵ تبدیل کند. البته این برنامه را می‌توان برای تبدیل تاریخ فیلم‌ها، کتاب‌های قدیمی و ساختمان‌های باستانی نیز به کار برد.

آرایه A را برای نگهداری ارزش رقم‌های رومی (M,D.C.L.X.V.I) به کار برده‌ایم. آرایهٔ C$ خود رقم‌های رومی

را دربردارد، هنگامی که یک رقم رومی عدد مورد نظر با آرایه C$ جور شد، ارزش آن رقم از خانهٔ معادل آرایهٔ A برداشته می‌شود.

```
100 REM CONVERTS ROMAN TO DECIMAL
110 DIM A (12,8),C$(8)
120 REM DATA FOR ARRAY A FOLLOWS
130 REM IF NO READ/DATA USE MULTIPLE LETS
140 DATA 100002,50003,10009,5005,1010,507,111
150 DATA 100002,50003,10009,5005,1010,507,111
160 DATA −1,−1,10009,5005,1010,507,111
170 DATA −1,−1,10004,5005,1010,507,111
180 DATA −1,−1,−1,5006,1010,507,111
190 DATA −1,−1,−1,−1,1006,507,111
200 DATA −1,−1,−1,−1,−1,508,111
210 DATA −1,−1,−1,−1,−1,−1,108
220 DATA 80005,30005,10004,5006,1010,508,111
230 DATA −1,−1,8007,3007,1006,508,111
240 DATA −1,−1,−1,−1,800,300,108
243 M = 0
250 REM LOAD ARRAY A
260 FOR J = 1 TO 11
270 FOR K = 1 TO 7
280 READ A(J,K)
290 NEXT K
300 NEXT J
310 REM DATA FOR ARRAY C$
320 DATA "M","D","C","L","X","V","I"
330 FOR J = 1 TO 7
340 READ C$(J) : NEXT J
350 REM GET ROMAN NUMBER
360 PRINT
370 INPUT "ROMAN NUMBER TO CONVERT";R$
380 FOR J = 1 TO LEN(R$)
390 T$ = LEFT$ (R$,1)
400 R$ = RIGHT$ (R$,LEN(R$)−1)
410 FOR K = 1 TO 7
420 IF T$ = C$(K) THEN 460
430 NEXT K
440 PRINT "NOT A VALID ROMAN NUMBER"
450 GO TO 350
```

```
460 REM PICK ACCUMULATED VALUE FROM
       ARRAY A
470 V = A(R,K)
480 REM --1 INDICATES ERROR
490 IF V < 0 THEN 440
500 REM ACCUMULATE RESULT SO FAR
510 M = M + INT(V/100)
520 NEXT J
530 PRINT "DECIMAL EQUIVALENT IS ";M
540 PRINT "MORE"
550 INPUT A$
560 IF A$ = "YES" THEN 350
999 END
```

سطر ۳۹۰ رقم دست‌چپ عدد رومی را برمی‌دارد و سطر ٤۰۰ آن را از عدد می‌اندازد؛ از این‌رو، سطر ۳۹۰ می‌تواند دوباره رقم بعدی دست‌چپ را بردارد. این‌دو دستور در دستگاههای مختلف ممکن است تفاوت داشته باشند. (البته این برنامه کامل نیست و نقص‌هایی دارد. شما می‌توانید به‌عنوان تمرین آن را کامل کنید. م)

نمونهٔ ۱۰ — سرمایهٔ آینده

این برنامه روش محاسبهٔ سرمایهٔ آینده را در یک سرمایه گذاری با نرخ بهرهٔ ثابت نشان می‌دهد. سرمایه آینده را می‌توان باتوجه به‌سرمایهٔ نخستین، مدت و نرخ بهره از دستور زیر به‌دست آورد:

$$FV = PV(1 + i\%)^n$$

که در آن FV سرمایهٔ آینده، PV سرمایهٔ نخستین، i نرخ بهره، n مدت سرمایه گذاری است.

با دستور بالا تنها می‌توان سود آیندهٔ هر سرمایه را به‌دست آورد؛ حال اگر سود ماهانه پرداخت شود و این سود هم دوباره سرمایه گذاری شود، باید بهرهٔ این سودها را نیز به‌محاسبه افزود. ولی باید به‌یاد داشت که مدت سرمایه گذاری این سودها با یکدیگر تفاوت دارند. مثلاً اگر تمام مدت

سرمایه‌گذاری ۶ ماه باشد، مدتی که اولین سود در سرمایه‌گذاری شرکت خواهد کرد ۵ ماه، دومی ۴ ماه و... آخری صفر ماه خواهد بود. می‌توان سود آینده را برای هرمبلغ جداگانه به‌دست آورد و بعد همه را باهم جمع کرد. هرچند می‌توان برای این کار از دستور کلّی زیر استفاده کرد (سِری‌های هندسیِ جبر را که فراموش نکرده‌اید؟):

$$\text{Total} = \frac{((1+i\%)^{n}-1)}{i\%} \times P$$

برنامهٔ زیر برای محاسبهٔ دستور بالا به کار رفته است.

```
9600 REM   SUBROUTINE   TO   CALCULATE   COM-
     POUND INTEREST
9610 REM PAYMENT IN P
9620 REM NO OF PERIODS IN N
9630 REM INTEREST IN I
9640 REM RESULT IN R
9650 REM
9660 X = (1+I/100)**N-1
9670 R = X/(I/100)*P
9680 RETURN
```

نمونهٔ ۱۱ ــ اجاره یا خرید؟

گاهی برخی از دستگاههایِ کارخانه‌ها را می‌توان به‌جای خریدن اجاره کرد. اجاره کردن این خوبی را دارد که سرمایه نمی‌خواهد، ولی این بدی را هم دارد که پس از به‌سر رسیدن مدت اجاره باید آن را به‌صاحبش برگرداند. برعکس، خریدن دستگاه باعث می‌شود که آن دستگاه از آنِ خریدار شود، ولی برای این کار باید پولی را که می‌توان سرمایه‌گذاری کرد صرف خریدن آن نمود. در زیرروال زیر اجارهٔ سالانه، مدت قرارداد، ارزش دستگاه، هزینه‌های سالانه نگهداری آن و نرخ بهره (برای برآورد سود سرمایهٔ صرف شده) به‌برنامه داده می‌شود و از آن خواسته می‌شود که تعیین کند کدام کار را باید انجام داد، اجاره یا خرید.

```
9200 REM LEASE COST IN L
9210 REM LEASE TERM IN N
```

```
9220 REM CAPITAL COST IN C
9230 REM MAINTENANCE ETC IN M
9240 REM INTEREST RECEIVABLE IN I
9250 REM
9260 REM CALCULATE PRESENT VALUE
9270 REM OF CAPITAL + MAINT (TO B)
9280 REM
9290 X = M*((1-(1+I/100)**-N)/I/100)
9300 B = X+C
9310 REM CALCULATE PRESENT VALUE
9320 REM OF LEASE PAYMENTS (TO R)
9330 R = L*(1+(1-(1+I/100)**-(N-1))/I/100)
9340 IF B < L THEN PRINT "BUY"
9350 IF B > L THEN PRINT "LEASE"
9360 IF B = L THEN PRINT "EQUAL PV"
9370 RETURN
```

می‌توان زیرروال را به‌سادگی چنان گسترش داد که هزینهٔ واقعی اجاره یا خرید را، چاپ کرده یا نمایش دهد. (توجه نمائید که هزینهٔ اجاره خرید برای زمان حال محاسبه شده است.)

نمونهٔ ۱۲ — هزینهٔ تلفن

این برنامه یکی از زمینه‌هایی را که کامپیوتر می‌تواند برای بازرگانی‌های کوچک بسیار سودمند باشد، یعنی برآورد هزینه، نشان می‌دهد. توجه نمائید که نمی‌گوئیم کنترل هزینه، زیرا کار کنترل برعهدهٔ مدیران می‌باشد. ولی مدیریت هنگامی می‌تواند کارآمد باشد که آگاهی‌های لازم را در دست داشته باشد. کامپیوتر در فراهم آوردن این آگاهی‌های لازم برای تصمیم گیری، وسیلهٔ بسیار سودمندی است.

هرچند این برنامه کامل نمی‌باشد (به‌خصوص باتوجه به‌فهرست پیچیدهٔ نرخ‌های شرکت‌های تلفن، که هرروز هم ممکن است تغییر کنند)، ولی نشان می‌دهد که کامپیوتر با چه سرعتی می‌تواند هزینهٔ یک تلفن را برحسب مدّت، و بسته به‌این که در ساعت‌های گران، معمولی یا ارزان انجام شده باشد، و نیز این که از نظر فاصله، محلی، راه نزدیک یا راه دور باشد، محاسبه نماید.

این روال مدّت تلفن را، بسته به‌ساعت و فاصله، به«واحد هزینه‌ای ارتباط تلفنی» تبدیل کرده و بعد آن را در نرخ تلفن (که در متغیر R قرار گرفته است) ضرب می‌نماید تا هزینهٔ تلفن به‌دست آید.

```
4000 REM INITIALISE VARIABLES
4010 P = 0
4020 S = 0
4030 C = 0
4040 PRINT "CALL TYPE  , A, B OR C="
4050 PRINT "LOCAL, SHORT-HAUL  , LONG-HAUL"
4060 INPUT A$
4070 PRINT "CALL TIME OF DAY 1, 2 OR 3="
4080 PRINT "PEAK, STANDARD OR CHEAP"
4090 INPUT D
4100 REM
4110 IF D = 3 THEN C=1
4120 IF D = 2 THEN S=1
4130 IF D = 1 THEN P=1
4140 PRINT
4150 PRINT "DURATION OF CALL MM, SS"
4160 INPUT M, S
4170 T = M+S/60
4180 IF A$ = "B" THEN GO TO 4220
4190 IF A$ = "C" THEN GO TO 4240
4200 LET B = C*INT(1+T/8)+S*INT(1+T/2)+P*INT
     (1+T/1.5)
4210 GO TO 4250
4220 LET B = C*INT(1+T/2.4)+S*INT(1+T/.75)+P*INT
     (1+T/.5)
4230 GO TO 4250
4240 LET B = C*INT(1+T/.8)+S*INT(1+T/.267)+P*INT
     (1+T/.2)
4250 PRINT "COST IS " B*R
4260 RETURN
```

پیوست ۱
مجموعهٔ نشانه‌های اسکی

Hex	*Dec*	*Char*	*Hex*	*Dec*	*Char*	*Hex*	*Dec*	*Char*	*Hex*	*Dec*	*Char*
00	0	NUL	20	32	SP	40	64	@	60	96	`
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	$	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	'	47	71	G	67	103	g
08	8	BS	28	40	(	48	72	H	68	104	h
09	9	HT	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	|
1D	29	GS	3D	61	=	5D	93	]	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	_	7F	127	DEL

Hex = عدد شانزده شانزدهی   *Dec* = عدد دهدهی   *Char* = نشانه

پیوست ۲

حساب شانزده شانزدهی

جدول جمع

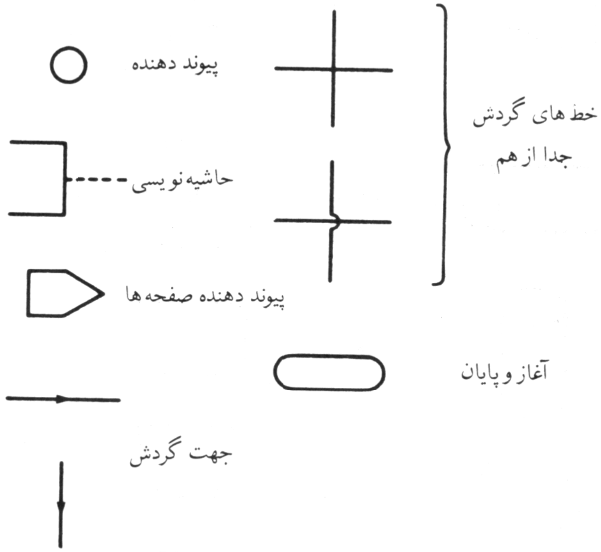
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
2	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
3	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12
4	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
5	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
6	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
7	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16
8	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17
9	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18
A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

جدول ضرب

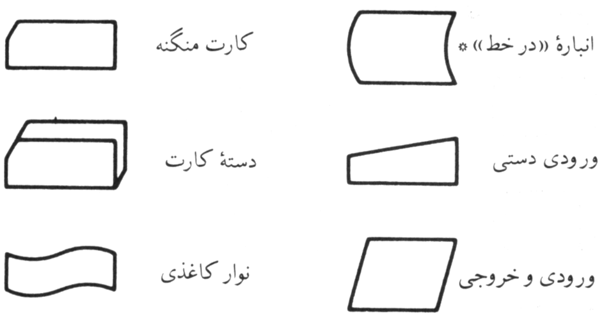
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
3	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
4	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

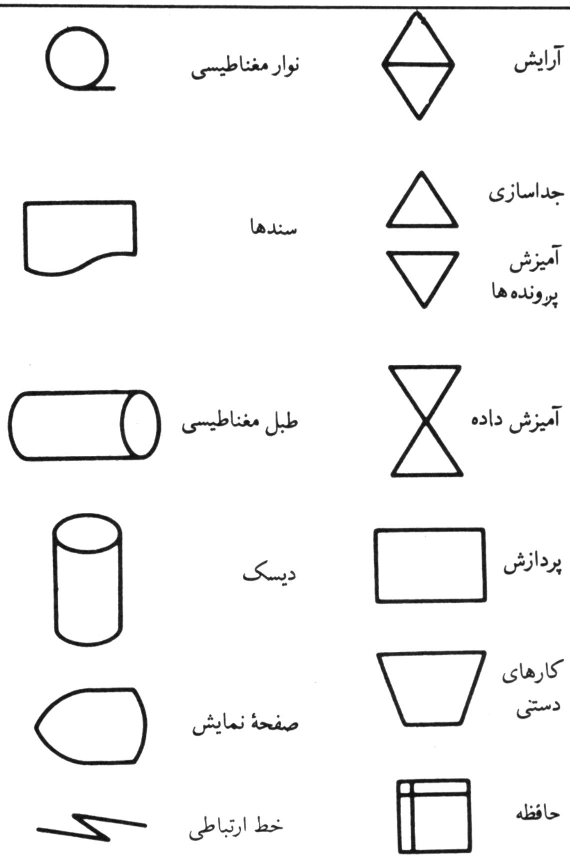
پیوست ۳
نمادهای گردش نمایی

نمادهای کلّی گردش نمایی

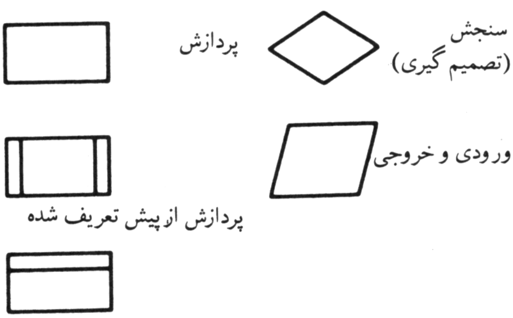


نمادهای گردش نمای سیستم





نمادهای گردش نمایی برنامه



پیوست ٤
توضیح اصطلاح ها

یادآوری مترجم: این پیوست را برای روشنتر شدن تعدادی از واژه‌ها و اصطلاح ها به کتاب افزوده‌ایم. برخی از این واژه‌ها و اصطلاح ها را ما خود ساخته‌ایم، زیرا یا برای آن‌ها معادلی پیش از این به کار نرفته بود، یا این که معادل به کار رفته را درست نمیدانستیم. البته در چند موردی هم شاید بتوان گفت به کار بردن واژه‌ای تازه برای معادلی جاافتاده، تنها به‌علت سلیقه مترجم بوده است و بس. امیدواریم که واژه‌های گویا و رسای پیشنهادی را همکاران دیگر نیز به کار گیرند و به‌جای واژه‌های نارسا و احتمالاً نادرست، واژه‌های درستتر و رساتر را پیشنهاد نمایند.

آرایش *(Sort)* : به کار آراستن و ترتیب دادن داده‌های یک آرایه یا پرونده، برحسب معیاری از پیش تعیین شده، آرایش گفته می‌شود. آرایش را می‌توان برحسب مقدار یک میدان، یا شماره‌ای معین و یا داده‌ای خاص انجام داد. مثلاً اگر اجزای یک آرایه مقدارهای عددی باشند، می‌توان آن را برحسب مقدار آن اجزاءِ به‌صورت فرازی (از کوچک به‌بزرگ)، یا فرودی (از بزرگ به کوچک) آراسته کرد. یا اگر پرونده‌ای دارای شماره کارمندی و نام کارمندان یک اداره باشد می‌توان آن را برحسب شماره کارمندی یا نام کارمندان مرتب کرد.

آرایش فرازی (یا صعودی) *(ascending order)* : نک: آرایش.

آرایش فرودی (یا نزولی) *(descending order)* :نک: آرایش.

آرایه *(array)* : آرایه را می‌توان همچون جدولی یک بُعدی، دو بعدی، یا چند بعدی در نظر گرفت که در هریک از خانه‌های آن مقداری عددی یا حرفی قرار گرفته باشد. هر خانهٔ این جدول را بسته به‌جای آن در

جدول، می‌توان با یک یا چند شماره مشخص کرد. این کاری است که در مورد آرایه انجام می‌شود و هنگام مراجعه به‌هریک از خانه‌های آن نام آرایه همراه یک یا چند شماره (بسته به‌تعداد بعدهای آرایه) به‌کار می‌رود. مثلاً اگر *A* آرایه‌ای یک بعدی با ده خانه باشد، هرخانه آن را می‌توان به‌شکل زیر نشان داد:

A(1), A(2), A(3), ... A(10)

اگر *B* آرایه‌ای دو بعدی و تعداد خانه‌های آن ۳×۱۰ باشد، آن‌ها را می‌توان به‌شکل زیر نشان داد:

A(1,1), A(1,2), A(1,3)
A(2,1), A(2,2), A(2,3)
A(10,1),,A(10,2), A(10,3)

هر خانهٔ آرایه را جزءِ آرایه نیز می‌نامیم. شماره‌ای را که در داخل پرانتز می‌نویسیم، زیرنویس می‌گوئیم

آسکی *(ASC II)* : آسکی که از حرف‌های نخستین واژه‌های عبارت

American Standard Code for Information Interchange

یعنی «رمز استاندارد آمریکایی برای بده و بستان‌های اطلاعاتی» ساخته شده است یکی از روش‌هایی است که برای نمودن نشانه‌ها به‌کار می‌رود. در این روش هر نشانه با ۷ بیت نشان داده می‌شود. مجموعهٔ نشانه‌های آسکی ۱۲۸ تا است که شامل حرف‌های کوچک و بزرگ، رقم‌های صفر تا ۹، نشانه‌های ویژه و نشانه‌های کنترلی می‌باشد. در زیر چند نشانه را با برابرهای شانزده شانزدهی آن‌ها نشان داده‌ایم:

A =۴۱ ،   a =۶۱ ،   * =۲A ،   + =۲B

در پیوست ۱، مجموعهٔ نشانه‌های اسکی را نشان داده‌ایم.

آماده ساختن آغازین *(Initialise)* : برای به‌کار بردن یک دیسکت

(یا دیسک) نو، باید پیش از هرکار آن را آمادهٔ پذیرفتن داده‌ها کرد. این کار که «آماده ساختن آغازین» نام دارد عبارت است از وارسی تمام کِشِه‌ها *(Tracks)*، تا کِشِه‌های عیب‌دار پیدا شده و به‌جای آن‌ها کِشِه‌های دیگر جایگزین شوند؛ و همچنین نوشتن داده‌های کنترلی، آماده کردن فهرست راهنما و نیز گذاردن نامی بر دیسکت.

انباره *(Storage)* : نک: حافظه

انبارهٔ درونی *(Internal Storage)* : نک: حافظه

انبارهٔ بیرونی *(External Storage)* : نک: حافظه

انبارهٔ درخط *(On – line storage)* : نک: درخط، حافظه

انبارهٔ دیسکی *(Disk storage)* : نک: دیسک مغناطیسی

بازگشت *(Return)* :۱ـ دکمهٔ بازگشت، دکمه‌ای است که هنگام دادن اطلاعات از صفحه کلید به‌ماشین، چون آن را فشار دهیم، اطلاعاتِ ماشین شده وارد دستگاه شده و برنامه کار را برروی آن آغاز می‌کند.

۲ـ دستور بازگشت. دستوری است که پایان زیرروال را نشان می‌دهد و با اجرای آن، کنترل به‌دستور بعد از «خواندن زیرروال» برمی‌گردد.

بازنشانی *(Reset)* : بازنشانی یعنی دادن مقدار آغازی به‌متغیّرها، مثلاً صفر کردن متغیرهای عددی و تهی کردن متغیرهای حرفی. در گفتگو از مدارها، بازنشانی به‌معنای برگرداندن مدار به‌حالت آغازی یا عادی می‌باشد، و در گفتگو از بیت‌ها به‌معنای تغییر بیت‌ها از ۱ به‌صفر. کلید بازنشانی دکمه‌ای است که چون فشار داده شده مدارها به‌حالت آغازی برگشته و دستگاه اطلاعات خود را از دست می‌دهد.

بازیابی *(Retrieval)* : به‌کار برداشتن یا خواندن داده‌های نگهداری شده در یک انباره، بازیابی گفته می‌شود.

بُرش *(Sector)* : نک: کِشِه

بُعدگذاری *(DIM = dimension)* : نک: آرایه

بیت *(Bit)* : بیت یا رقم دودویی *(BInary digit)* همان رقم‌های صفر و یک است که در عددهای دودویی به‌کار می‌رود. مثلاً، عدد دودویی ۱۰۰ (که برابر عدد ٤ دهدهی است) دارای سه بیت است که دوتای نخستین صفر و سومی یک می‌باشد.

بیسیک *(BASIC)* : بیسیک که از حرف‌های نخستین واژه‌های عبارت

Beginner's All—purpose Symbolic Instruction Code یعنی

«زبان برنامه‌نویسی همه‌کاره برای تازه‌کاران» ساخته شده است، زبانی است که در آغاز برای آموزش برنامه‌نویسی کامپیوتر به‌وجود آمد، ولی امروزه دارای کاربرد بسیار گسترده‌تری شده است. گونه‌های جدید این زبان بسیار کامل و پیشرفته می‌باشند.

پاسکال *(Pascal)* : یکی از زبان‌های برنامه‌نویسی بسیار پیشرفته می‌باشد که به‌نام فیلسوف و دانشمند نامی فرانسوی، پاسکال نامیده شده است.

پایانه *(Terminal)* : پایانه دستگاهی است که برای ارتباط مستقیم با کامپیوتر به‌کار می‌رود. این دستگاه معمولاً از یک صفحه کلید (شبیه ماشین تحریر) و یک صفحهٔ نمایش تلویزیونی یا دستگاه چاپ تشکیل شده است. پایانه اگر با کامپیوتر در یک محل قرار داشته باشد، دستگاه فرمان (کنسول) نامیده می‌شود، و اگر از کامپیوتر دور باشد، به‌آن پایانهٔ دوردست گفته می‌شود.

پرونده *(File)* : به‌مجموعه‌ای از داده‌های مربوط به‌هم که دارای نامی باشد پرونده گفته می‌شود، مثل پروندهٔ نام و نشانی کارمندان یک اداره. برای نگهداری پرونده می‌توان از کارت منگنه، نوار مغناطیسی و یا دیسک استفاده کرد.

پروندهٔ پیوسته *(Sequential file)* : پروندهٔ پیوسته پرونده‌ای است که داده‌ها در آن پشت سرهم و پیوسته به‌یکدیگر قرار گرفته باشند و در میان

آن‌ها فاصله‌ای نباشد. چنانچه بخواهیم به‌این پرونده‌ها دادهٔ تازه‌ای افزوده شود، باید آن را به‌پایان پرونده بیافزاییم.

پشته *(Stack)* : مجموعه‌ای از داده‌ها را که تنها از یک طرف آن بتوان دادهٔ تازه‌ای به‌آن افزود، یا از آن برداشت، پشته می‌نامیم. پشته را می‌توان همچون ستونی از داده‌ها در نظر گرفت که هر دادهٔ تازه به‌بالای ستون افزوده می‌شود، و هر داده‌ای که برداشته می‌شود از بالای آن برداشته می‌شود. از این رو آخرین دادهٔ افزوده شده نخستین داده‌ای است که برداشته می‌شود. این روش خواندن و نوشتن داده‌ها را پشته‌سازی می‌نامیم.

پشته‌سازی *(Stacking)* : نک: پشته

پی ال وان *(PL1)* : زبان برنامه‌نویسی گسترده‌ای است که هم در کارهای علمی و هم در کاربردهای بازرگانی می‌توان آن را به‌کار برد.

تراشه *(Chip)* : تراشه عبارت است از تکه نازک و کوچک مستطیل شکل سیلیسیمی (مثلاً حدود ۵ میلی‌متر مربع) که قطعه‌های مداری یک نیمه رسانا را بر روی آن کار گذاشته‌اند.

ثابت *(Constant)* : به‌پارامتری که مقدار آن به‌هنگام اجرای برنامه تغییر نکند ثابت گفته می‌شود. به‌ثابت‌هایی که دارای نشانه‌های حرفی یا حرف و عددی باشد رشته نیز گفته می‌شود. نک: رشته

جابه‌جایی *(Shift)* : نک: دکمهٔ جابه‌جایی

جهش *(Branch , Jump)* : جهش دستوری است که جریان عادی اجرای برنامه را تغییر می‌دهد. یعنی به‌جای این که دستورها پشت سرهم اجراءِ شوند، بعد از اجرای دستور جهش، کنترل برنامه به‌دستور دیگری منتقل می‌شود. دستور خاص این کار در بیسیک دستور *GO TO* است که به‌آن دستور جهش غیرشرطی نیز گفته می‌شود. چون پس از اجرای این دستور، کنترل برنامه بی‌درنگ به‌دستوری، کم‌وبیش دور از آن منتقل می‌شود. افزون براین دستور، دستور دیگری به‌نام جهش شرطی هم هست که هنگامی کنترل

را منتقل می‌کند که شرط خاصّی برقرار باشد. مثلاً *IF A>10 THEN 120*

جهش شرطی *(Conditional branch Jump)* : نک: جهش

جهش غیرشرطی *(Unconditional branch Jump)* : نک: جهش

حافظه *(Memory)* : حافظه، که گاهی به‌آن انباره *(Storage)* نیز گفته می‌شود، وسیله یا دستگاهی است که می‌تواند داده‌ها را دریافت کرده و در خود نگهداری نماید تا دوباره مورد استفاده قرار گیرند. حافظه بردو گونه است:

۱— حافظهٔ اصلی (یا انبارهٔ درونی) که جزئی از خود کامپیوتر است و برای نگهداری سیستم عامل و برنامه به کار می‌رود.

۲— حافظهٔ کمکی (یا انبارهٔ بیرونی) که در بیرون کامپیوتر جای گرفته است، همچون دیسک، نوار مغناطیسی و جز آن. چنانچه انبارهٔ بیرونی مستقیماً برای پردازنده قابل دسترسی باشد، گوییم «درخط» قرار دارد، و از این رو آن را «انبارهٔ درخط» می‌نامیم.

اندازهٔ حافظه را با «ک» (*K* = کیلو) نشان می‌دهیم. هر «ک» برابر ۱۰۲٤ واحد است اگر حافظهٔ دستگاهی ۱٦ ک باشد، اندازهٔ آن ۱٦٫۳۸٤ واژه می‌باشد.

حافظهٔ اصلی *(Main Memory)* : نک: حافظه

حافظهٔ کمکی *(Auxiliary Memory)* : نک: حافظه

حلقه *(Loop)* : به‌مجموعهٔ چند دستور برنامه که بارها تکرار می‌شوند حلقه گفته می‌شود. حلقه معمولاً با برقرار شدن شرط خاصی به‌پایان می‌رسد، مثلاً با ۱۰۰ بار تکرار شدن.

خانهٔ آرایه *(element)* : نک: آرایه

خروجی *(out put)* : خروجی نتیجهٔ پردازش است که به‌بیرون فرستاده می‌شود. خروجی ممکن است به‌صورت چاپ، نمایش تصویری و یا انتقال داده‌ها برروی یک انبارهٔ بیرونی، مثل دیسک یا نوار مغناطیسی،

باشد.

خود فراخوانی *(Recursion)* : چنانچه زیرروال توسط خود آن فراخوانده شود آن زیرروال را «خود فراخوان» و این روش را «خود فراخوانی» می‌نامیم.

داده‌ها *(Data)* : داده‌ها عبارتند از اطلاعاتی که دربارهٔ حقیقت‌ها، مفهوم‌ها و چیزهای دیگر جمع‌آوری شده و به‌شیوه‌ای خاص، و برای منظوری خاص، سازماندهی یافته‌اند.

درخط *(on-line)* : نک: حافظه

درگاه *(Port)* : وسیله‌ای است که برای گرفتن و دادن داده‌ها از یک دستگاه به‌کار می‌رود. درگاه معمولاً رشته سیمی است که در هر سر آن چندین شاخک سوزنی شکل قرار دارد. هریک از این شاخک‌ها برای اتصال مداری به‌مدار دیگر به‌کار می‌روند.

درون‌داد *(Input)* : نک: ورودی

دستگاه نمایش *(Display device)* : دستگاهی است شبیه تلویزیون که دارای صفحه‌ای برای نشان دادن داده‌ها می‌باشد. از این دستگاه برای نمایش دادن خروجی‌ها استفاده می‌شود. صفحه‌ای را که داده‌ها بر روی آن نشان داده می‌شوند، صفحهٔ نمایش *(Screen)* می‌نامیم.

دکمهٔ بازگشت *(Return Rey)* : نک: بازگشت.

دکمهٔ بازنشانی *(Reset key)* : نک: بازنشانی

دکمهٔ جابه‌جایی *(Shift key)* : با هر دکمهٔ صفحه کلید می‌توان دو نشانه (مثلاً شکل‌های کوچک و بزرگ یک حرف خاص) را ماشین کرد. یکی از این‌دو را می‌توان با فشار دادن دکمه ماشین کرد ولی برای ماشین کردن دوّمی باید همزمان دکمهٔ جابه‌جایی را نیز گرفت. پس دکمهٔ جابه‌جایی برای تغییر حرف‌ها از کوچک به‌بزرگ — و در بعضی از دستگاهها از بزرگ به‌کوچک — و همچنین انتخاب یکی از دو نشانهٔ هر کلید به‌کار

می‌رود.

دکمهٔ گریز *(Escape)* : از دکمه‌های صفحه کلید علاوه‌بر ماشین کردن حرف‌ها و نشانه‌ها، می‌توان برای دادن نشانه‌های کنترلی به‌دستگاه استفاده کرد. برای این کار باید هنگام فشار دادن هر دکمه، همزمان دکمهٔ گریز را نیز گرفت. نشانه‌های کنترلی نشانه‌هایی هستند که بیشتر برای تغییر کار سخت‌افزار به‌کار می‌روند. کار این دکمه ممکن است در دستگاههای مختلف یکسان نباشد، از این‌رو بهتر است در این‌مورد به‌کتاب راهنمای دستگاه مراجعه شود.

دودویی *(Binary)* : دستگاه عددیی که پایهٔ آن دو باشد دودویی نام دارد. این دستگاه تنها رقم‌های صفر و یک را دارا می‌باشد. از آنجا که صفر و یک را می‌توان با خاموش و روشن کردن یک چراغ، یا آهنربا کردن یک حلقه در یکی از دو جهت، نشان داد، تنها این دستگاه عددی است که در کامپیوتر و دستگاههای وابسته به‌آن به‌کار می‌رود. در زیر چند عدد دودویی را با عددهای دهدهی برابر با آن‌ها نشان داده‌ایم:

عدد دهدهی	عدد دودویی	عدد دهدهی	عدد دودویی
٤	۱۱۰	۰	۰
۵	۱۰۱	۱	۱
۱۰	۱۰۱۰	۲	۱۰
۱۰۰	۱۱۰۰۱۰۰	۳	۱۱

ارزش هر رقم در دستگاه دودویی از ضرب آن رقم در ارزش مکانی آن به‌دست می‌آید. ارزش‌های مکانی به‌ترتیب از راست به‌چپ عبارتند از: ۱، ۲، ٤، ۸، ۱٦، ۳۲، ٦٤، ۱۲۸

دیسک مغناطیسی *(Magnetic disc)* : انباره‌ای است که برای نگهداری داده‌ها و دستیابی مستقیم به‌آن به‌کار می‌رود. دیسک از یک یا چند صفحه مغناطیسی (شبیه صفحهٔ گرام) ساخته شده است. اطلاعات

به‌شکل ذرّه‌های مغناطیس شده برروی این صفحه‌ها ثبت می‌شوند. این اطلاعات را دوبار می‌توان از روی دیسک، به‌وسیلهٔ کامپیوتر خواند.

رشته *(String)* : به زنجیره‌ای از نشانه‌های حرفی، عددی یا حرف و عددی (که نشانه‌های ویژه را نیز دربر می‌گیرد) رشته می‌گوئیم. رشته‌ای که تنها از حرف تشکیل شده باشد رشتهٔ حرفی، و رشته‌ای که از حرف، عدد و نشانه‌های ویژه ساخته شده باشد، رشتهٔ حرف و عددی نام دارد.

رشتهٔ حرفی *(Literal string)* : نک: رشته

رشتهٔ حرفی و عددی *(Alphanumeric string)* : نک: رشته

ریز پردازنده *(Microprocessor)* : مدار مجتمع یا تراشه‌ای است که برای انجام دادن دستورهایی، و نیز سرپرستی و کنترل کارهای یک دستگاه هوشمند، مانند کامپیوترهای کوچک و مُدِم‌ها *(Modem)* به‌کار می‌رود. (به‌دستگاههای تلفیق و تفکیک کننده‌ای که برای دادن اطلاعات تلفنی به‌کامپیوتر از آن‌ها استفاده می‌شود، مُدِم گفته می‌شود.)

ریز کامپیوتر *(Microcomputer)* : نک: کامپیوتر خانگی

زبان هدف *(Object language)* : زبان هدف زبانی است که از ترجمه یک زبان برنامه‌نویسی (مثلاً بیسیک) به‌دست می‌آید. زبان هدف معمولاً همان زبان ماشین می‌باشد. زبان ماشین تنها زبانی است که کامپیوتر می‌تواند مستقیماً آن را دریابد، از این‌رو زبان‌های دیگر نخست باید به‌زبان ماشین ترجمه شوند تا کامپیوتر بتواند آن‌ها را بفهمد. زبان نخستین را که به‌زبان هدف ترجمه می‌شود زبان مبدأ می‌نامیم.

زبان ماشین *(Machine language)* :نک: زبان هدف

زبان مبدأ *(Source language)* : نک: زبان هدف

زیرروال *(Subroutine)* :زیرروال عبارت است از چند دستور که باهم کار خاصّی را انجام می‌دهند و در برنامه‌های گوناگون مورد استفاده قرار می‌گیرند، یا یک برنامه آن‌ها را چندین‌بار مورد استفاده قرار می‌دهد.

زیرروال‌ها معمولاً در روی دیسکت یا نوار مغناطیسی، در جایی به‌نام کتاب‌خانهٔ زیرروال‌ها، نگهداری می‌شوند. این زیرروال‌ها را می‌توان در برنامه‌های گوناگون به کار برد.

زیرنویس *(subscript)* : نک: آرایه

سیستم عامل *(operating system)* : مجموعهٔ چند برنامه که برای سرپرستی و کنترل کار دستگاه، توسط سازندگان کامپیوتر، فراهم آمده‌اند. همهٔ کارهایی که کامپیوتر انجام می‌دهد با رهبری و سرپرستی سیستم عامل می‌باشد. پس از روشن کردن دستگاه و پیش از هرکار دیگر باید سیستم عامل را وارد کامپیوتر کرد. چنانچه سیستم عامل برروی دیسک نگهداری شود به‌آن سیستم عامل دیسکی گفته می‌شود.

سیستم عامل دیسکی *(disc operating system)* : نک: سیستم عامل

شانزده شانزدهی *(Hexadecimal)* : دستگاهی است عددی که پایهٔ آن ۱٦ می‌باشد و ارقام آن عبارتند از صفر تا ۹ و *A* تا *F* . ارزش مکانی هر رقم، برابر توانی از عدد ۱٦ و به‌ترتیب زیر می‌باشد:

رقم نخست عدد	= ۱×رقم	زیرا	۱=$۱٦^{۰}$
رقم دوم عدد	= ۱٦×رقم	زیرا	۱٦=$۱٦^{۱}$
رقم سوم عدد	= ۲۵٦×رقم	زیرا	۲۵٦=$۱٦^{۲}$
رقم چهارم عدد	= ٤۰۹٦×رقم	زیرا	٤۰۹٦=$۱٦^{۳}$

این دستگاه را همچنین برای نشان دادن عددهایی دودویی به‌کار می‌برند، زیرا به‌کاربردن آن بسیار ساده‌تر است. برای این کار هرچهار رقم عدد دودویی را (ازدست‌راست) با یک رقم شانزده شانزدهی نشان می‌دهیم. مثال:

۱۰۱۱،۰۰۱۱،۱۱۰۰،۱۱۱۱ = *B۳CF*

صافی کوک شده *(Tuned filter)* : صافی کوک شده که به‌نام **صافی تراز شده** *(Balanced filter)* نیز نامیده می‌شود، مداری است که

دارای چند خازن و مقاومت می‌باشد. کار این صافی آن است که بعضی از بسامدها را تقویت کرده و بسامدهای دیگر را از میان بردارد. برای مثال می‌توان یک صافی را چنان تنظیم کرد که تنها بسامدهای یک کیلوهرتز و کمتر را تقویت کند و بسامدهای دیگر را نابود سازد، یا تنها بسامدهای ۲ تا ۳ کیلوهرتز را تقویت نماید.

صفحهٔ نمایش *(screen)* : نک: دستگاه نمایش

فرمان‌های خانه‌داری *(Housekeeping commands)*

فرمان‌های خانه‌داری معمولاً به‌فرمان‌هایی گفته می‌شود که روی انباره‌های بیرونی کاری انجام می‌دهند، همچون باز کردن و بستن پرونده‌ها، آماده ساختن آغازین، زدودن پرونده‌ها و تغییر نام آن‌ها.

فهرست گزینه‌ها *(Menu)* : فهرست گزینه‌ها فهرست یا لیستی است که کارهای مختلفی را که یک برنامه می‌تواند انجام دهد برروی صفحهٔ نمایش نشان می‌دهد. استفاده کننده می‌تواند هرکدام را که بخواهد انتخاب کرده و با فشار دادن دکمه‌ای خاصّ، یا دادن شماره‌ای به‌دستگاه، از کامپیوتر بخواهد که آن کار را انجام دهد. گاهی بعد از انتخاب یک گزینه، فهرست دیگری پدیدار می‌شود که باید از آن بازگزینهٔ دیگری را برگزید و این کار ممکن است چند بار تکرار شود.

کار همزمانی *(Time sharing)* : به‌کار چند فرد که در یک زمان از یک کامپیوتر استفاده می‌کنند کار همزمانی گفته می‌شود. در کار همزمانی هر استفاده‌کننده چنان با کامپیوتر کار می‌کند که گویی دستگاه تنها در اختیار او قرار دارد و استفاده‌کنندهٔ دیگری در کار نیست.

کامپیوتر خانگی *(Home computer)* : کامپیوتر خانگی، که به‌آن کامپیوتر شخصی و ریز کامپیوتر نیز گفته می‌شود، دستگاه ارزان قیمتی است که برای کارهای سرگرمی و نیز بازرگانی‌های کوچک ساخته شده ست. این دستگاه بر پایهٔ ریز پردازنده قرار گرفته و معمولاً دارای یک صفحهٔ

کلید، یک دستگاه نمایش (یا وسیلهٔ استفاده از تلویزیون معمولی)، حدود ٦٤ ک حافظه اصلی، و حافظهٔ کمکی به‌شکل کاست یا دیسکت می‌باشد. زبان‌هایی که معمولاً در کامپیوترهای خانگی به‌کار می‌روند عبارتند از بیسیک، پاسکال و اسمبلر.

کامپیوتر شخصی *(Personal computer)* : نک: کامپیوتر خانگی

کِشِه *(Track)* : در دستگاههایی که داده‌ها را به‌صورت ذرات مغناطیسی درمی‌آورند، مثل دیسکت و نوار کاست، به‌خط های باریکی که سه‌صدم اینچ پهنا دارند، و داده‌ها برروی آن‌ها نوشته می‌شوند، کِشه گفته می‌شود. این کِشه‌ها در دیسک مغناطیسی، و نیز دیسکت، به‌صورت دایره‌هایی هم‌مرکز می‌باشند. هر دایره را به‌چند کمان بخش کرده‌اند که به‌آن‌ها بُرش *(Sector)* گفته می‌شود.

کلید بازنشانی *(Reset Key)* : نک: بازنشانی

کوبول *(COBOL)* : کوبول که از حرف‌های نخستین واژه‌های عبارت

Common Business– Oriented Language

یعنی «زبان همگانی کارهای بازرگانی» ساخته شده است زبان برنامه‌نویسی است که بیشتر در کاربردهای بازرگانی و صنعتی به‌کار می‌رود.

گردش‌نما *(Flowchart)* : نمایش تصویری آنچه را که می‌خواهیم با کامپیوتر انجام دهیم، و نیز ترتیب اجرای این کار را، گردش‌نمای آن کار می‌نامیم. گردش‌نمای برنامه، کار یک برنامه و مراحل آن را نشان می‌دهد. گردش‌نمای سیستم، کار یک کاربرد خاص را ــ که معمولاً از چند برنامه تشکیل می‌شود ــ و جریان اطلاعات را در آن سیستم نشان می‌دهد. کار استفاده از گردش‌نما را «گردش‌نمایی» می‌نامیم.

گردش‌نمای برنامه *(Program flowchart)* : نک: گردش‌نما

گردش‌نمای سیستم *(System flowchart)* :نک: گردش‌نما

مترجم *(Compiler)* : مترجم، که گاهی به‌آن مؤلف نیز گفته می‌شود، برنامه‌ای است که توسط سازندگان کامپیوتر فراهم شده است و کار آن ترجمهٔ برنامه از زبان مبدأ به‌زبان هدف می‌باشد. زبان مبدأ ممکن است یکی از زبان‌های بیسیک، کوبول، پاسکال و جز آن باشد. زبان هدف معمولاً همان زبان ماشین است.

مجموعهٔ نشانه‌ها *(Character set)* : نک: نشانه

مفسّر *(Interpreter)* :مفسّر برنامه‌ای است که توسط سازندگان کامپیوتر فراهم آمده و کار آن ترجمهٔ دستورهای زبان مبدأ (مثلاً بیسیک) و اجرای آن‌ها می‌باشد. تفاوت مفسر و مترجم در این است که مترجم نخست همهٔ برنامه را به‌زبان هدف (یا زبان ماشین) ترجمه کرده و سپس آن را اجراءِ می‌کند؛ ولی مفسر هر دستور را یکی یکی ترجمه و اجراءِ می‌نماید. در حالت نخست برنامه یک‌بار ترجمه شده و بعد از آن می‌توان از برنامهٔ ترجمه شده استفاده کرد ولی در حالت دومِ هر دستور برنامه باید قبل از اجراءِ دوباره ترجمه شده و ترجمهٔ آن هم نگهداری نمی‌شود.

مکان‌نما *(Cursor)* : مکان‌نما خط‌تیرهٔ چشمک‌زنی است که در هنگام ماشین کردن نشانه‌ها، جایی را که نشانهٔ بعدی در روی صفحه نمایش خواهد گرفت، نشان می‌دهد.

نرم‌افزار *(Software)* : نرم‌افزار عبارت است از برنامه‌هایی که برای اجرا شدن در یک کامپیوتر فراهم گشته‌اند. این برنامه‌ها یا به‌وسیلهٔ سازندهٔ دستگاه تهیّه شده‌اند (مانند سیستم عامل)، و یا به‌دست استفاده‌کنندگان دستگاه فراهم آمده‌اند (مانند برنامه‌های معمولی).

نشان *(Flag)* : نشان را می‌توان متغیری در نظر گرفت که می‌تواند دو مقدار یا دو ارزش را نشان دهد. یکی از این دو ارزش نشان‌دهندهٔ بودن شرط

یا حالتی، و ارزش دیگر نشان دهندهٔ نبودن آن شرط یا حالت است. برای نمونه می‌توان عدد ۱ را برای بودن شرط و عدد صفر را برای نبودن شرط به کار برد. نشان نادرستی نشانی است که برای نمایش اشتباه یا نادرستی داده یا کاری خاص از آن استفاده می‌شود.

نشان نادرستی *(Error flag)* : نک: نشان

نشانگر *(Pointer)* : نشانگر را می‌توان همچون متغیری در نظر گرفت که مقدار آن نشان‌دهندهٔ نشانی محلی در حافظه می‌باشد. مثلاً در دستور «خواندن» زبان بیسیک، نشانگر خاصی داده‌ای را که باید خوانده شود نشان می‌دهد. هربار که این دستور اجراءِ می‌شود بر مقدار نشانگر چنان افزوده می‌شود که با دستور خواندن بعدی، دارهٔ بعدی در دسترس قرار گیرد.

نشانه *(Character)* : نشانه عبارت است از شکل یا ترتیبی از اجزاءِ (خط، نقطه...) که برای آن معنی خاصی را در نظر گرفته‌ایم؛ و این معنی را می‌توان از دیدن آن نشانه دریافت کرد. مثل نشانه‌های چاپی یا نوشتاری (حرف‌ها، رقم‌ها و...) و نشانه‌های نگاره‌ای که در رسم نمودارها به کار می‌روند. نشانه‌های نوشتاری (یا به‌طور خلاصه نشانه‌ها) عبارتند از حرف‌های A تا Z ، رقم‌های ۰ تا ۹، و نشانه‌های ویژه همچون +، /، ؟، *، .... این نشانه‌ها روی هم مجموعهٔ نشانه‌های یک دستگاه را می‌سازند. در پیوست ۱ مجموعهٔ نشانه‌های اسکی را به‌دست داده‌ایم.

نشانهٔ تهی *(Space/Blank character)* : نشانهٔ تهی نشانه‌ای است که در هنگام چاپ چیزی برای آن چاپ نمی‌شود ولی به‌اندازهٔ جای یک نشانه نانوشته باقی می‌ماند. نشانهٔ تهی را در حقیقت می‌توان در هنگام چاپ برای فاصله گذاشتن میان نشانه‌های دیگر به کار برد. این نشانه همچون نشانه‌های دیگر در حافظهٔ کامپیوتر و انباره‌های بیرونی جایی را می‌گیرد.

نشانهٔ نگاره‌ای *(Graphic character)* : نک: نشانه

نشانهٔ نوشتاری (یا چاپی) *(Printed character)* : نک: نشانه

نشانهٔ ویژه *(Special character)* : نک: نشانه

واژهٔ اندوخته *(Reserved word)* : در بیشتر زبان‌های برنامه‌نویسی، برخی واژه‌ها دارای معنای خاصی می‌باشند و نباید به‌وسیلهٔ برنامه‌نویسی برای کار دیگری به‌کار برده شوند. مثلاً در زبان بیسیک واژه‌های *IF* و *READ* از این واژه‌ها هستند.

واگذاری *(Assignment)* : واگذار کردن یعنی دادن مقداری تازه به‌یک متغیر. این مقدار می‌تواند عددی یا حرفی باشد؛ و نیز نتیجه یک عبارت ریاضی باشد یا از به‌هم پیوستن چند رشتهٔ حرفی به‌دست آید. در زبان بیسیک دستور واگذاری با علامت مساوی (یعنی =) مشخص می‌شود. افزون برآن می‌توان در آغاز دستور واژهٔ *LET* را نیز قرار داد. مثال: *LET X = Y + Z*

ورودی *(INPUT)* : ۱— ورودی یا درون‌داد عبارت از داده‌ای است که به کامپیوتر داده می‌شود تا برروی آن کار ویژه‌ای را انجام دهد. همچنین به کار دادن داده‌ها به دستگاه نیز درون داد گفته می‌شود.

۲— در زبان بیسیک نام دستوری است که داده‌ای را از عبارت داده‌ها برداشته و به برنامه می‌دهد.

٦٤ ک حافظه *(64 K Memory)* : نک: حافظه

واژه‌نامه ۱

فارسی به انگلیسی

آراستن ۱۵۷، ۱۵۸	Sort
آرایش ۱۷۵، ۱۰۱	Sort
آرایش لایه‌ای ۱۵۸	Shell sort
آرایش موجی ۱۰۱	Ripple sort
آرایه ۱۷۵، ۹۲	Array
آزمایش ۲۲	Test
آشکارا ۱۴۵	Explicity
آماده‌سازی آغازین ۱۷۶، ۶۹	Initialise
آمیختن ۱۵۸	Merge
آمیزش ۱۵۹	Merge
آمیزش پرونده‌ها ۱۷۴	Collate

PASCAL	پاسکال ۲۱،۱۷۸
Clear	پاک کردن ۳۷
Rubout	پاک کن ۱٤۸
Terminal	پایانه ۲۰،۱۷۸
Process	پردازش کردن
Processing	پردازش ۱۷٤
Processor	پردازنده ۱۰
File	پرونده ٦۷،۸٤،۱۷۸
Sequential file	پروندهٔ پیوسته ۱۷۸
Stack	پُشته ۱۱۱،۱۷۹
Stacking	پشته‌سازی ۱۱۲،۱۷۹
Backup	پشتیبانی (کردن) ۷۲
Null	پوچ ۹۰
Pl/1	پی ال وان ۲۱،۱۷۹
Scanner	پیماینده
Scan	پیمودن (آرایه) ۹٦
Sequential	پیوسته ٦٦،۸٦
Connector	پیوند دهنده ۱۷۳
Function	تابع ۱۱۷
Chip	تراشه ۱۰،۱۷٦
Decision	تصمیم گیری ۱۷٤
Nested	تودرتو(حلقه های) ٤٦،۵۳
Nesting	تودرتوسازی (زیرروال ها) ۱۱۱
Constant	ثابت ۲۸،۱۷۸
Numeric constant	ثابت عددی ۲۸

درست‌سازی داده‌ها ١٣٨،١٠٨	Data validation
درشت‌نگاشت (نگاره) ١٣٥	Low Resolution
درگاه ١٨١،٦٣،٩	Port
درگاههای ورودی و خروجی ٩	Input output ports
درون داد ١٨١،٧٥	Input
درونی (تابع) ١٢٢	Intrinsic
دستگاه نمایش ١٨١،٩	Display unit
دستور	Instruction
دکمهٔ بازگشت ١٨١،١٤	Return key
دکمهٔ بازنشانی ١٨١،١٥	Reset key
دکمهٔ جابه‌جایی ١٨١،١٥	Shift key

دکمهٔ گریز ١٨٢،١٥	Escape key
دودویی ١٨٢،٦٢	Binary
دیسک/دیسکت ١٨٢،٦٦	Disc/diskette
دیسکت گردان ٨٤	Disc Drive

راندن ٢٨،١٦	Run
رشته ١٨٣،٢٨	String
رشتهٔ حرف و عددی ١٨٣،٢٨	Alphanumeric string
رشتهٔ حرفی ١٨٣،٢٨	Literal string
روال ١٦٥،١١٤	Routine
روز آمد کردن ٧١	Update
رهبری کردن ٧٠	Administer,control
ریخت ٢٢	Format
ریز پردازنده ١٨٣،١٠	Microprocessor

Integer	عدد صحیح ۲۸،۱۱۷
Call	فراخواندن ۱۱٤
Ascending	فرازی (آرایش) ۱۰۱،۱۷۵
Command	فرمان ۱۵،۲۳
Descending	فرودی (آرایش) ۱۰۱،۱۷۵
High level	فشرده (گردش‌نما) ۱۳۹
Double-density	فشردگی دوگانه ٦۷
Menu	فهرست گزینه‌ها ۸۰،۱۸۵
Lock	قفل کردن ٦۹
Unlock	قفل گشایی (دستور) ۷۰
Application	کاربرد ۲۰،۱۵۷
Punched card	کارت منگنه ۹
Time sharing	کار همزمانی ۲۰،۱۸۵
Home computer	کامپیوتر خانگی ۹،۱۸۵
Personal computer	کامپیوتر شخصی ۱۰،۱۸٦
Minicomputer	کامپیوتر کوچک ۱۰
Track	کِشه ٦۷،۱۸٦
Control	کنترل ٤۹
Action	کُنِش ٤۷
COBOL	کوبول ۹،۲۱،۱۸٦
Tuned	کوک شده ٦۵
Flowchart	گردش‌نما ٤٦،۱۳۹،۱۸٦
Flowcharting	گردش‌نمایی ٤٦،۱۷٤،۱۸٦

نشانهٔ ویژه ۱۸۸	Special character
نقطه ویرگول ۳۳،۸۲	Semicolon
نگارش ۴۹،۷۲	Version
نگاره زدایی (دستور) ۱۲۸	Unplot
نگاره سازی ۱۲۷	Graphics
نگاره گذاری (دستور) ۱۲۸	Plot
نماینده ۳۵	Indicator
نمودار گروهی ۱۳۹	Block diagram
نمودار میله‌ای ۱۳۲	Bar chart
نوار مغناطیسی ۶۱،۶۵	Magnetic tape
نوای شنیداری ۶۵	Audio tone
وارسی	Control
واژه ۳۵	Word
واژهٔ اندوخته ۳۵،۱۸۹	Reserved word
واگذار کردن ۲۹،۱۸۹	Assigm
واگذاری ۲۹،۱۸۹	Assignment
ورودی ۹،۱۸۹	Input
همزمان ۷۰	Immediate
همساز ۶۷	Compatible
همخوانی داشتن ۷۷	Match
یاختهٔ تصویری ۱۲۷	Pixel
یادآوری ۱۴۱	Remark

Stop	ایست ۴۳
Storage	انباره ۶۱
Store	انبار کردن ۶۱
String	رشته ۲۸،۱۸۳
Subroutine	زیرروال ۱۰۴،۱۸۳
Subscript	زیرنویس ۹۲،۱۸۴
Tabulation	جدول‌بندی ۳۶
Terminal	پایانه ۲۰،۱۷۸
Test	آزمایش ۲۲
Textual variable	متغیّر حرفی ۳۲،۷۶
Time sharing	کار همزمانی ۲۰،۱۸۵
Track	کِشِه ۶۷،۱۸۶
Tuned	کوک شده ۶۵
Tuned filter	صافی کوک شده ۶۵،۱۸۴
Unconditional branch	جهش غیرشرطی ۴۳،۱۸۰
Unlock	قفل کشایی (دستور) ۷۰
Unplot	نگاره زدایی (دستور) ۱۲۸
Update	به‌هنگام درآمدن، زورآمد کردن ۷۹
Value	ارزش ۲۵ مقدار ۲۹
Varible	متغیّر ۲۹
Version	نگارش ۴۹،۷۲ گونه ۳۱
Word	واژه ۳۵

Reset	بازنشانی ۱۵،۱۷۷
Reset key	دکمهٔ بازنشانی ۱۵،۱۸۱
Retreival	بازیابی ٦٦،۱۷۷
Return	بازگشت ۱٤،۱۰۵،۱۷۷
Return key	دکمهٔ بازگشت ۱٤،۱۸۱
Ripple sort	آرایش موجی ۱۰۱
Routine	روال ۱۱٤،۱٦۵
Rubout	پاک کن ۱٤۸
Run	راندن ۱٦،۲۸
Scan	پیمودن (آرایه) ۹٦
Scanner	پیماینده
Screen	صفحهٔ نمایش ۱٤،۱۸۵
Sector	بُرش ٦۷،۱۷۷
Semicolon	نقطه ویرگول ۳۳،۸۲
Sequential	پیوسته (پرونده) ۱۷۸
Sequential file	پروندهٔ پیوسته ۱۷۸
Shell sort	آرایش لایه‌ای ۱۵۸
Shift key	دکمهٔ جابه‌جایی ۱۵،۱۸۱
Silicon	سیلیسیم ۱۰
Software	نرم افزار ۹،۱۸۷
Sort	آراستن ۱۵۸،۱۷۵
Source	خاستگاه،مبداءِ (زبان) ۲۱،۱۸۳
Space character	نشانهٔ تهی ٤٦،۱۸۸
Special character	نشانهٔ ویژه ۱۸۸
Stack	پشته ۱۱۱،۱۷۹
Stacking	پشته سازی ۱۲۲،۱۷۹

Intrinsic	درونی (تابع) ۱۲۲
Jump	جهش ۴۰،۴۳،۱۷۹
List	لیست، لیست گرفتن ۱۶،۱۴۷
Literal string	رشتهٔ حرفی ۲۸،۱۸۳
Load	بار کردن ۶۶
Lock	قفل کردن ۶۹
Loop	حلقه ۲۲،۱۸۰
Low level	گسترده (گردش‌نمای) ۱۳۹
Low resolution	درشت نگاشت (نگاره) ۱۳۵
Machin languge	زبان ماشین ۲۱،۱۸۳
Magnetic tape	نوار مغناطیسی ۶۱،۶۵
Main memory	حافظهٔ اصلی ۱۸۰
Match	جور بودن ۳۸ همخوانی داشتن ۷۷
Matching	جور کردن
Memory	حافظه ۹،۱۸۰
Menu	فهرست گزینه‌ها ۸۰،۱۸۵
Merge	آمیختن ۱۵۸ آمیزش ۱۵۹
Microprocessor	ریز کامپیوتر ۱۰،۱۸۳
Microcomputer	ریز پردازنده ۱۰،۱۸۳
Minicomputer	کامپیوتر کوچک ۱۰
Nested	تودرتو(حلقه‌های) ۴۶،۵۳
Nesting	تودرتوسازی (زیرروال‌ها) ۱۱۱
Nul	پوچ ۹۰
Numeric constant	ثابت عددی ۲۸،۱۷۸
Numeric variable	متغیّر عددی ۲۹
Object language	زبان هدف ۲۱،۱۸۳

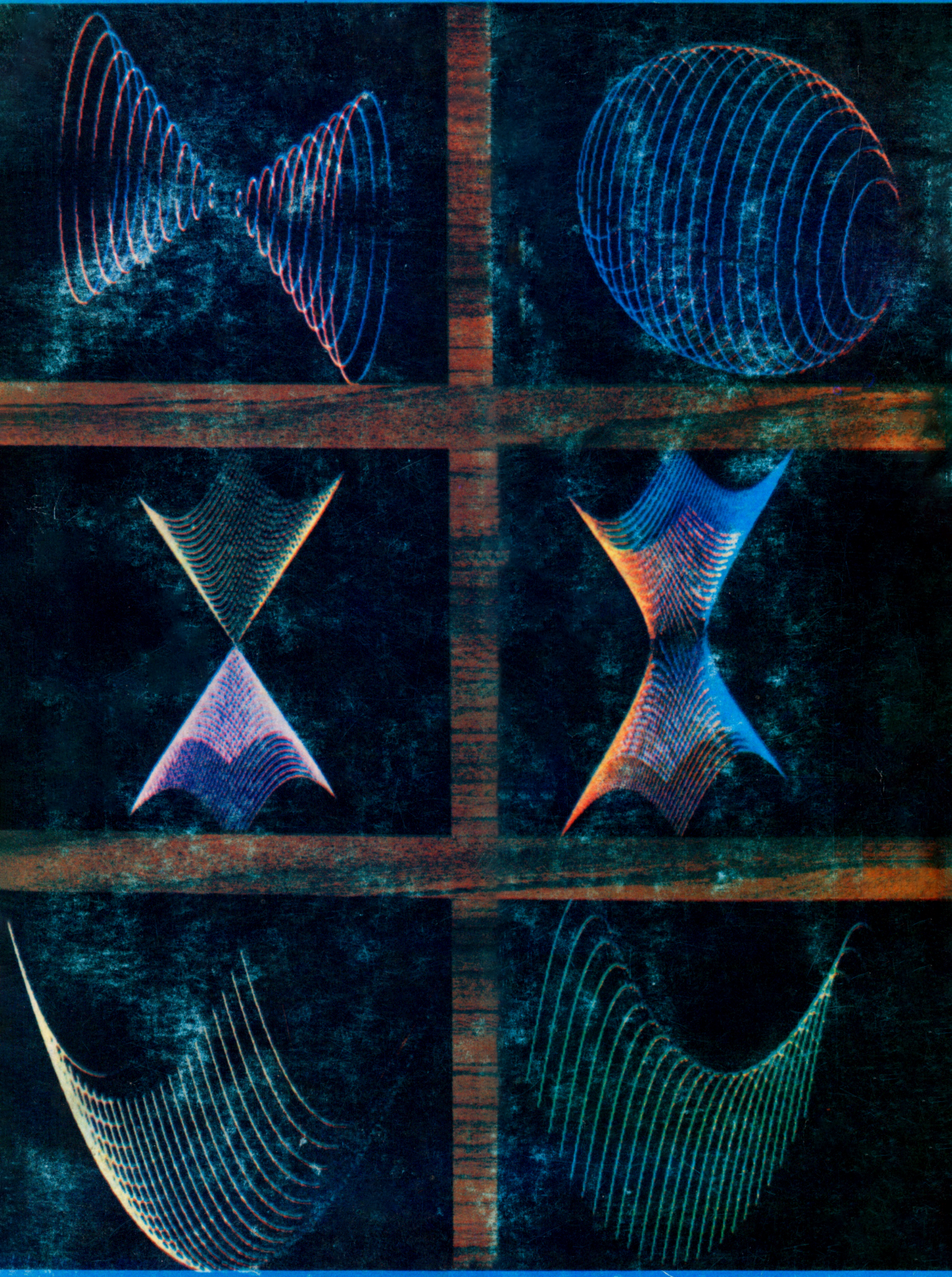
Data	داده ها ۳۷،۱۸۱
Data validation	درست‌سازی داده ها ۱۰۸،۱۳۸
Decision	تصمیم گیری ۱۷٤
Delete	زدودن ٦۹
Depending GOTO	جهش وابسته ۵۸
Descending	فرودی (آرایش) ۱۰۱،۱۷۵
Dialect	گویش ۲۰
Dimension	بُعد، بُعد گذاری ۹٦،۱۷۷
Disc diskette	دیسک/دیسکت ٦٦،۱۸۲
Disc drive	دیسک گردان ۸٤
Disc operating system	سیستم عامل دیسکی ۹،٦۷،۱۸٤
Disc storage	انبارهٔ دیسکی ۹،۱۷۷
Display unit	دستگاه نمایش ۹،۱۸۱
Document	سند
Documentation	مستند سازی ۱۳۷
Documented	مستند شده ۱۳۷
Double- density	فشردگی دوگانه ٦۷
Drum	طبل (مغناطیسی) ۱۷٤
Dummy	ساختگی (شناسه) ۱۲۳
Element	خانه، جزءِ (جدول، آرایه) ۹۲،۱۸۰
Error flag	نشان نادرستی ۹۱،۱٤٦،۱۸۸
ESCAPE key	دکمهٔ گریز ۱۵،۱۸۲
Explicitly	آشکارا ۱٤۵
External storage	انبارهٔ بیرونی ٦۵،٦٦،۱۷۷
Graphic character	نشانهٔ نگاره‌ای ۱۸۸

واژه‌نامه ۲

انگلیسی به فارسی

Action	کنش ۴۷
Administer	رهبری ۷۰
Alphanumeric String	رشتۀ حرف و عددی ۲۸،۱۸۳
Annotation	حاشیه‌نویسی ۱۷۳
Application	کاربرد ۲۰،۱۵۷
Argument	شناسه ۱۲۴
Array	آرایه ۹۲،۱۷۵
Ascending	فرازی (آرایش) ۱۰۱،۱۷۵
ASC II	آسکی ۹۰،۱۷۱،۱۷۶
Assign	واگذار کردن ۲۹،۱۸۹
Assignment	واگذاری ۲۹،۱۸۹
Audio tone	نوای شنیداری ۶۵

بهاء ۳۶۰ ریال



زبان برنامه‌نویسی BASIC برای کامپیوترهای کوچک

انتشارات بعثت